Université des Sciences et des Technologies de Lille
École Doctorale Sciences Pour l'Ingénieur

Thèse de Doctorat

Spécialité : **Informatique**

présentée par
Merwan BARLIER

Sur le rôle de l'être humain dans le dialogue humain/machine

sous la direction de **Olivier PIETQUIN**
et l'encadrement de **Romain LAROCHE**

Soutenue publiquement à **Villeneuve d'Ascq**, le **14 décembre 2018** devant le jury composé de :

M. Matthieu **GEIST**	Google Brain	Président du jury
Mme. Kristiina **JOKINEN**	AIST AI Research Center	Rapporteur
M. Fabrice **LEFEVRE**	Université d'Avignon	Rapporteur
M. Philippe **PREUX**	Université de Lille	Examinateur
M. Olivier **PIETQUIN**	Google Brain	Directeur de thèse
M. Romain **LAROCHE**	Microsoft Research	Encadrant industriel

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL), UMR 9189,
Équipe SequeL, 59650, Villeneuve d'Ascq, France

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse et ami, Olivier Pietquin, pour m'avoir accompagné tout au long de cette thèse. Sa bonne humeur, sa disponibilité, son enthousiasme, sa vision des choses et sa patience ont été les fondements de la réussite de ce travail. En dehors de l'aspect scientifique, ces quatre années passées en sa compagnie m'ont été à la fois très enrichissantes mais également extrêmement rigolotes.

Je tiens également à remercier Romain pour m'avoir encadré à Orange. Ses perspectives, sa rigueur et sa vivacité d'esprit ont également permis l'accomplissement de ces travaux.

Je voudrais ensuite remercier Kristiina Jokinen et Fabrice Lefèvre pour avoir accepté d'être rapporteurs de cette thèse. Mes remerciements vont également à Matthieu Geist et Philippe Preux pour avoir accepté de faire partie du jury.

Je souhaiterais également remercier tous les doctorants, chercheurs et stagiaires d'Orange et de SequeL avec qui j'ai pu partager mon quotidien : Hatim, Julien, Florian, Alex, Xuedong, Nicolas, Mathieu, Ronan, Michal, Giorgos et tous les autres en nombre trop important pour pouvoir être cités.

En dehors du travail, je souhaiterai également remercier tous les copains : Gus, Nico, Nico, Chapo, Caro, Matthieu, Mathieu, Bouli, David, JB, Geo, Arnaud et tous les autres.

Je voudrais également dire un grand merci à Dominique pour m'avoir accueilli avec toute sa gentillesse et sa générosité au cours de ces trois premiers mois de thèse.

Je souhaiterai également dire merci à toute ma famille pour son soutien durant cette thèse.

Enfin, un grand merci à Laurie pour avoir été à mes côtés et m'avoir soutenu tout au long de cette dernière année.

Table des matières

Chapitre 1
Introduction

Introduction

La parole est le moyen de communication le plus naturel de l'être humain. Partant de ce constat, il est raisonnable d'imaginer que l'interaction vocale avec les ordinateurs est l'un des moyens les plus efficaces pour bénéficier au maximum de leurs capacités. Le milieu de l'intelligence artificielle saisit très vite les promesses d'un ordinateur capable de dialoguer et très tôt, le dialogue humain/machine fut élevé au rang d'application iconique du domaine. En effet dès l'aube de l'informatique, alors que les ordinateurs n'en étaient qu'à leurs balbutiements, Alan Turing, l'un des premiers penseurs et acteurs du domaine, suggéra de valider le succès de la réalisation d'une intelligence artificielle par une procédure dialogique dans laquelle un humain cherche à déterminer si son interlocuteur est un être humain ou un ordinateur (Turing (1950)).

Cette envie de concevoir une machine capable de dialoguer avec les êtres humains amena dans les années 1960 à ELIZA, le premier système de dialogue conçu par l'homme impliquant une machine simulant les réponses d'un psychothérapeute à un patient en lui reposant ses questions (Weizenbaum (1966)). La Figure 1.1 [1] illustre le type de dialogue obtenus avec ce système. Quelques années plus tard, PARRY, un système similaire, fut conçu pour étudier la schizophrénie (Colby et al. (1971)).

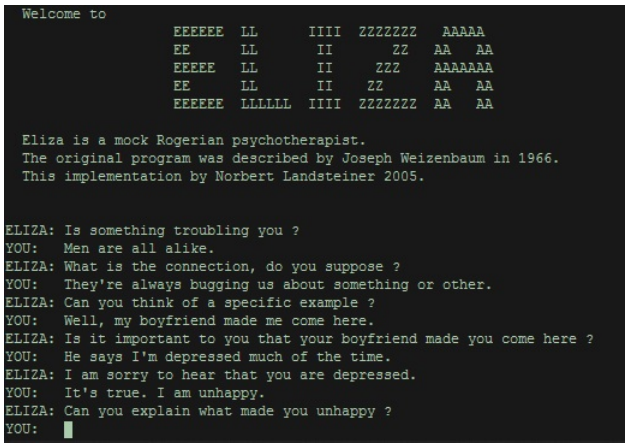


Figure 1.1 – Système ELIZA

1. Image tirée de la page Wikipédia anglais du système ELIZA : https://en.wikipedia.org/wiki/ELIZA

De nouvelles architectures telle GUS (Bobrow et al. (1977)) permirent depuis de concevoir des systèmes de dialogue plus généraux, capables de converser sur des taches couvrant un périmètre restreint. Malheureusement, ces systèmes n'étaient capables de dialoguer que dans un unique contexte et étendre leur champ d'utilisation impliquait de recommencer tout le travail préalable à leur implémentation. Ce n'est que à partir de la fin des années 1990 que le dialogue humain/machine fut abordé comme un problème de décision dans l'incertain avec l'utilisation des processus décisionnels de Markov Levin and Pieraccini (1997). Ce changement de paradigme ouvrit la voie aux approches statistiques, ce qui permit un bond technologique, permettant de concevoir des systèmes de plus en plus puissants.

Ce n'est que très récemment que des systèmes interactifs basés sur la voix furent adoptés par le grand publique (Bellegarda (2013)). Ainsi de nombreuses initiatives industrielles telles Echo, Siri, Cortana ou Google Now, présentées en Figure 1.2, ont profité des avancées significatives de la recherche dans le domaine de l'apprentissage automatique et dans ses applications en reconnaissance vocale et en compréhension du langage naturel pour démocratiser la parole en tant que médium pour l'interaction humain/machine, proposant une alternative au traditionnel trio clavier/souris/écran. Ces systèmes, bien que se basant sur la parole, ne sont cependant pas des systèmes de dialogue, car ils se basent sur le paradigme de la commande vocale, éventuellement combiné avec des algorithmes de recherche. Ici, les réponses proposées par le système se rapportent à chaque question de l'utilisateur mais aucune référence au passé (ou peu) n'est prise en compte dans le choix et la construction des réponses.

FIGURE 1.2 – Assistants vocaux

Si la recherche dans les systèmes de dialogue a pu se développer au delà de la commande vocale, c'est de part les problématiques dues à la dimension séquentielle de la tâche. En effet, chaque énoncé du système de dialogue a des conséquences sur le futur de la conversation, que ce soit à court, moyen ou long terme. Ainsi, bien que des modules de compréhension et de génération de la parole sont nécessaires dans le cas de la commande vocale et dans celui du dialogue, il est possible de faire avancer la recherche dans les systèmes de dialogue en supposant résolues ces deux problématiques. Dans ce cas, l'étude se porte sur le module responsable de l'interaction dialogique, appelé le *gestionnaire de dialogue*. C'est celui-ci que nous étudions dans ce manuscrit.

Un problème de décision dans l'incertain

La tâche de la prise de décision séquentielle est généralement traitée selon le paradigme agent/environnement présenté en Figure 1.3.

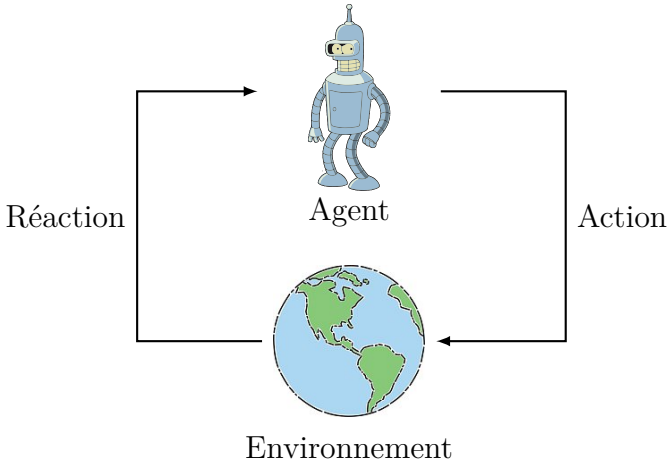


FIGURE 1.3 – Modèle agent-environnement

Selon ce paradigme, à chaque pas de temps[2], un *agent*, *i.e.* une entité capable de percevoir l'*environnement* dans lequel il se trouve, interprète l'état de celui-ci à partir de l'information fournie par ses *capteurs* et agit par le biais de ses *actuateurs*. Son action change alors l'état de l'environnement, ce qui entraîne une réaction de la part de celui-ci. Cette réaction est alors observée par l'agent, ce qui relance le cycle.

L'ensemble des perceptions passées de l'agent est appelé *historique*. Celui-ci est généralement agrégé en un *état*. C'est à partir de cet état, et uniquement à partir de lui, que l'agent décide quelle action accomplir. Dans le cadre du dialogue humain/machine, l'environnement est principalement composé de l'interlocuteur de la machine. Les signaux perçus par la machine sont, quant à eux, les sorties de son module de compréhension. L'état de la machine consiste donc en une représentation de l'avancée du dialogue, menant à une représentation de la connaissance et de l'objectif de son utilisateur.

L'ensemble des approches existant pour définir la fonction, appelée *politique*, reliant les états aux actions de l'agent, peut se situer sur un spectre dont, à une extrémité se trouvent les politiques implémentées par un expert humain, "pensant" à la place de la machine. On parle dans ce cas d'une *politique à base de règles*. À l'opposé du spectre, on a système qui, au fur et à mesure de ses interactions avec l'environnement, définit sa propre politique par lui-même. On parle alors de politique *apprise*.

La Figure 1.4 résume les différents modules mis en oeuvre au cours de la prise de décision d'un agent suite à un signal envoyé par son environnement.

2. Pour les problèmes à temps continu, l'agent peut agir à n'importe quel moment.

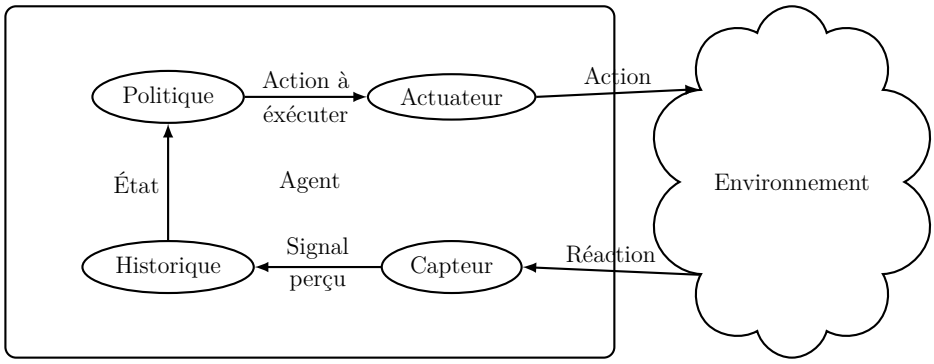


FIGURE 1.4 – Architecture d'un agent

Bien que les premiers succès de la recherche en dialogue furent dus à des systèmes implémentés à base de règles, cette approche mena rapidement à une impasse. Deux raisons expliquent principalement ces limites. Tout d'abord, penser à la place de la machine implique d'être capable de construire un système de règles complet statuant de manière irréfutable quelle action choisir au bon moment. Construire un tel système est une tâche difficile et chronophage. En effet, ces règles doivent être en nombre suffisant pour apporter une réponse juste dans tous les cas de figure. Elles doivent également être cohérentes entre elles, une tâche d'autant plus dure que le nombre de règles est grand. Par ailleurs, comme nous allons le développer au cours de cette thèse, l'un des besoins majeurs est de concevoir des systèmes évolutifs, aptes à faire face à la non-stationnarité du monde. Concevoir des systèmes de dialogues à base de règles est donc un problème difficile, car en adoptant cette approche, il est nécessaire de "connaître" cette non-stationnarité.

Inversement, un système basé sur une politique apprise est capable d'agir de manière autonome. Il peut tout d'abord saisir l'état de son environnement pour ensuite planifier une séquence d'actions permettant de réaliser ces objectifs au mieux. Pour ces systèmes de type *apprenant*, il est souvent commode de transcrire ces objectifs sous la forme d'une récompense, dans ce cas l'agent cherchera à maximiser une fonction cumulative de ses récompenses futures.

Un tel agent doit être capable de développer des capacités de planification afin de trouver la meilleure séquence d'actions nécessaires pour accomplir son objectif. Celle-ci est généralement fonction de l'environnement dans lequel se trouve l'agent.

Une grande catégorie de systèmes de dialogue, qu'ils soient académiques ou industriels, ont pour principal objectif la résolution d'une tâche. Selon ce paradigme, le dialogue est vu comme un problème épisodique dont chaque épisode correspond à une tentative de résolution de la tâche.

Dans ce cadre, la politique apprise par un système de dialogue est indépendante de son interlocuteur. Celui-ci est en effet censé se comporter au mieux quel que soit son utilisateur. Les facultés cognitives des êtres humains sont cependant très variables. Un adolescent baignant dans un bassin informatique depuis son plus jeune âge n'aura par

exemple pas d'appréhension à dialoguer avec une machine, tandis qu'une personne âgée non familière avec les ordinateurs pourra être beaucoup plus réticente. Pour pallier cette difficulté, la variabilité de comportement des utilisateurs est encodée dans la stochasticité de l'environnement. Sont également encodés dans celle-ci les erreurs de transcription de la reconnaissance vocale ou bien des erreurs de compréhension du message transcrit. Ce problème de l'apprentissage d'une prise de décision séquentielle dans un environnement incertain est appelé *apprentissage par renforcement* et est utilisé avec succès dans les systèmes de dialogue depuis son introduction dans Singh et al. (2000).

Limites du cadre actuel

Le modèle d'apprentissage par renforcement exposé précédemment peut cependant ne pas suffire pour faire la transition des systèmes de dialogue orientés tâches vers des systèmes de dialogue de type assistants personnels.

Premièrement, l'une des contraintes à prendre en compte est que l'aspect épisodique de la tâche disparaît pour ce type de système. En effet, bien que les tâches que cherche à résoudre un utilisateur à l'aide de son système de dialogue sont a priori indépendantes, il est attendu de la part de ce système une prise en compte de l'ensemble des dialogues passés pour extraire les caractéristiques les plus pertinentes de son utilisateur et le servir au mieux.

Dans un second temps, il est important de prendre en considération les facultés d'adaptation de l'être humain. En effet, au fur et à mesure des interactions, son comportement se modifie dans le but d'accomplir ses objectifs de manière plus efficace. Cette modification comportementale a de nombreuses implications sur les spécificités du système de dialogue, qui doit être à même de s'adapter à un agent s'adaptant lui-même, et de prendre en compte ces modifications comportementales pour atteindre ses objectifs.

Ces deux phénomènes de co-adaption sont présents dans la majorité des interactions humain/machine. Un parallèle peut être fait avec l'exemple du e-learning (ou *curriculum learning*), où la machine cherche à faire apprendre quelque chose à un être humain, comme par exemple une langue étrangère. Dans cet exemple, l'adaptation de l'être humain est le phénomène recherché, on désire qu'au fur et à mesure des interactions, il progresse dans son apprentissage. La machine doit donc continuellement modifier ses exercices pour lui faire apprendre plus.

Suivre les progrès de l'utilisateur humain peut dans ce cadre se faire par le biais de contrôles régulièrement posés. Intégrer les résultats de ceux-ci à la représentation de l'environnement permet à la machine de gérer la non-stationnarité du comportement de l'utilisateur. Par ailleurs, en étudiant l'évolution de ces résultats au cours du temps, il est possible d'avoir une idée de l'évolution de son apprentissage. En disposant d'un ensemble de données présentant différentes courbes d'apprentissages types, la machine pourra assigner à son utilisateur l'un de ces types pour permettre une meilleure compréhension de celui-ci.

Cette approche n'est malheureusement pas applicable au dialogue humain/machine. On peut en effet difficilement imaginer un système de dialogue soumettant régulièrement un questionnaire à son utilisateur. Par ailleurs, les informations recherchées par le système de dialogue sont plus difficilement quantifiables.

Enfin, définir l'objectif d'un assistant personnel et l'encoder sous la forme d'une fonction de récompense représentent une nouvelle difficulté. Dans le cas d'un dialogue orienté tâche, il est aisé de définir une fonction objectif attestant la réussite de la tâche alors que dans le cas d'un assistant personnel, cette notion de réussite n'existe pas nécessairement.

Passer d'un système de dialogue orienté tâches à un système de type assistant personnel requiert donc une extension du cadre de l'apprentissage par renforcement. C'est pourquoi la thèse soutenue dans ce manuscrit est qu'il faut aller au-delà du traditionnel modèle agent/environnement. Pour cela, différentes manières de tenir compte de l'être humain dans les systèmes de dialogue sont analysées.

Contributions

Dans la section précédente, nous avons brièvement abordé la problématique de l'apprentissage par renforcement. Ce problème sera formellement abordé en Section 2.2. L'apprentissage par renforcement est un domaine de l'apprentissage automatique se posant la question de comment développer un système capable d'adapter son comportement dans un environnement donné afin d'accomplir au mieux ses objectifs, encodés sous la forme d'une certaine fonction de récompense à maximiser.

Ainsi qu'exposé préalablement, depuis Singh et al. (2000), le problème du dialogue humain/machine est généralement abordé comme un problème d'apprentissage par renforcement. Dans ce cadre, la fonction de récompense est définie par le concepteur de la machine. Les objectifs du système de dialogue sont donc définis par un être humain cependant, la manière d'atteindre ces objectifs n'est pas spécifiée et doit être découverte par la machine.

Comme statué dans la section précédente, l'apprentissage par renforcement se place dans un cadre agent/environnement. Dans ce cadre, l'agent n'est conçu pour être efficace que dans un environnement stationnaire et dénué d'objectif. Un système de dialogue conçu dans cette optique ne pourra donc être efficace que si son utilisateur ne modifie pas son comportement au cours du temps et si le dialogue est orienté tâche, requérant que la machine et son utilisateur coopèrent afin d'accomplir les objectifs de l'utilisateur. Ces deux hypothèses sont cependant aisément violées. Pour pallier ces problèmes, nous proposons dans notre première contribution, Barlier et al. (2015), de modéliser le dialogue humain/machine par un jeu stochastique. Les avantages d'un tel modèle sont illustrés par une preuve de concept dans laquelle l'apprentissage par renforcement échoue, contrairement à l'apprentissage par les jeux stochastiques.

L'apprentissage par renforcement est par ailleurs une boite noire par laquelle il est difficile d'obtenir le comportement désiré d'un système en modifiant manuellement ses

paramètres. L'utilisateur d'un système de dialogue ne peut donc pas corriger celui-ci lors de ses erreurs, ou encourager ses bons comportements. Cet utilisateur doit donc sans cesse attendre la prochaine mise à jour du système pour espérer voir une amélioration, qui ne concernera pas nécessairement les points les plus importants. Un système de prise en considération de conseils humains, encodés sous la forme d'une fonction de récompense alternative, est présentée dans Barlier et al. (2018). Dans cette contribution, il est proposé d'implémenter un système apprenant capable de prendre en considération un ensemble de conseils par l'utilisation d'une architecture de type TAMER. Par ailleurs, la sortie commerciale d'un système de dialogue ne peut se faire que sous l'assurance que le système se comportera de manière convenable. Obtenir des garanties sur un tel système est généralement coûteux, notamment en termes de nombre de données nécessaires à l'apprentissage. Il est montré dans ce travail que le cadre proposé permet également de réduire de manière significative cet apprentissage.

Un système de dialogue de type assistant personnel devra être capable de gérer des situations ne pouvant se mettre sous la forme d'un dialogue orienté tâche. L'une d'entre elles est étudiée dans Barlier et al. (2016a). Ici, il est considéré une situation dans laquelle un système de dialogue doit intervenir pour améliorer l'issue d'un dialogue humain/humain. Ce problème peut être rencontré dans une multitude de contextes. Celui mis en lumière dans cette contribution est celle de deux humains dialoguant dans l'objectif de prendre un rendez-vous, la machine écoute la conversation et est capable d'intervenir avec l'objectif de donner des informations pertinentes à son utilisateur. Il est attendu que ces informations aient le moins d'impact possible sur la conversation, un compromis doit donc être trouvé entre modifier profondément le cours de la conversation, au risque d'être trop intrusif, et ne pas intervenir. Il est proposé dans Barlier et al. (2016a) de gérer ce compromis en introduisant une pénalité mesurant la dissimilarité entre la conversation telle qu'elle aurait dû se produire et la conversation modifiée. Incorporer une telle pénalité permet alors de ramener le problème à un problème d'optimisation de processus décisionnels de Markov linéaires, problème ayant été longuement étudié.

Dans la précédente contribution, il est proposé de voir le dialogue humain/humain assisté par ordinateur comme un problème d'optimisation de processus décisionnels de Markov linéaires. Cependant, un tel cadre implique de connaître la dynamique d'une conversation humain/humain. Ce problème est récurrent dans l'apprentissage par renforcement. Dans Barlier et al. (2016b), nous proposons de l'aborder par l'étude de l'apprentissage d'automates en se basant sur la méthode des moments.

Plan du manuscrit

Ce manuscrit s'articule autour de deux parties. La Partie I est consacrée aux systèmes de dialogue et à l'apprentissage automatique. Dans le Chapitre 2, nous rappelons les bases de l'apprentissage supervisé en mettant l'accent sur la régression linéaire et les réseaux de neurones. Nous étudions ensuite les principaux algorithmes de l'apprentissage par renforcement. Dans le Chapitre 3, nous détaillons l'architecture des systèmes

de dialogue et nous voyons en quoi les algorithmes vus dans le chapitre précédent sont utiles dans la conception de tels systèmes.

La Partie II développe les contributions apportées à cet état de l'art par cette thèse. Dans cette partie, nous présentons trois scénarios différents dans lesquels les modèles traditionnels d'apprentissage par renforcement ne sont pas directement applicables. Dans le Chapitre 4, nous présentons une situation dans laquelle l'être humain s'adapte à son système de dialogue de manière à être le plus efficace possible, situation illustrée par un dialogue de négociation. Dans le Chapitre 5, nous abordons un scénario dans lequel l'humain fournit des conseils à son système de dialogue pour permettre une optimisation de celui-ci plus rapide et plus sûre. Une dernière situation est abordée au Chapitre 6 : le dialogue humain/humain assisté par ordinateur. Dans celui-ci, un système appelé Assistant de Conversation écoute les dialogues de son utilisateur, son objectif est d'intervenir lorsqu'il estime que son intervention sera profitable pour celui-ci et qu'elle ne perturbera pas trop la conversation. Une approche de calcul de distributions de probabilité, nécessaire à l'implémentation de la méthode précédente est ensuite développée dans le Chapitre 7.

Finalement, le Chapitre 8 clôt ce manuscrit en présentant de futures pistes de recherche.

Liste des publications

Les travaux réalisés au cours de cette thèse ont donné lieu aux publications suivantes dans des conférences internationales :

- BARLIER, Merwan, PEROLAT, Julien, LAROCHE, Romain, et PIETQUIN, Olivier. Human-machine dialogue as a stochastic game. In : *Proceedings of SigDial*, 2015.
- BARLIER, Merwan, LAROCHE, Romain, et PIETQUIN, Olivier. A stochastic model for computer-aided human-human dialogue. In : *Proceedings of Interspeech*, 2016.
- BARLIER, Merwan, LAROCHE, Romain, et PIETQUIN, Olivier. Learning dialogue dynamics with the method of moments. In : *Proceedings of SLT*, 2016.
- BARLIER, Merwan, LAROCHE, Romain, et PIETQUIN, Olivier. Training Dialogue Systems With Human Advice. In : *Proceedings of AAMAS*, 2018.

Première partie

Apprentissage automatique et systèmes de dialogue

Chapitre 2
Apprentissage automatique

Russell et al. (1995) définissent l'apprentissage d'un agent par l'utilisation de sa connaissance du monde qui l'entoure pour améliorer son comportement dans celui-ci. Dans ce chapitre, nous allons présenter dans un premier temps des méthodes, appelées méthodes d'*apprentissage supervisé*, permettant à l'agent d'utiliser l'information extraite dans des situations connues pour la généraliser à des situations inconnues. Dans un second temps, nous verrons, à travers une présentation de l'*apprentissage par renforcement*, de quelle manière ledit agent peut apprendre à améliorer son comportement au cours du temps.

2.1 Apprentissage supervisé

2.1.1 Définition

L'apprentissage supervisé (*Supervised Learning* en anglais) est une branche de l'apprentissage automatique dont le rôle est d'apprendre la fonction reliant des entrées à des sorties à partir d'un ensemble d'exemples. Formellement, supposons que l'on dispose d'une base de données $\mathcal{D} = \{x^i, y^i\}_{i=1}^N$, où à chaque variable x^i, appelée *entrée*, correspond une étiquette y^i, appelée *sortie*. À partir de ce jeu de données, l'objectif est alors de retrouver la fonction f, appelée *prédicteur*, assurant la correspondance entre les données et les étiquettes. L'intérêt de ce problème réside dans le fait que, connaissant le prédicteur, il est possible d'associer une sortie à chaque entrée non étiquetée.

Lorsque les sorties sont quantitatives, la tâche d'apprentissage supervisée est qualifiée de *régression*, tandis que lorsqu'elles sont qualitatives, on parlera de tâche de *classification*.

À partir d'un jeu de données uniquement, il n'existe pas d'algorithme capable de retrouver exactement le prédicteur dans tous les cas possibles. Un nouvel objectif se définit alors : plutôt que de rechercher parmi l'ensemble des fonctions possibles le "vrai" prédicteur, on va chercher parmi une classe de fonctions, appelée *ensemble d'hypothèses*, celle généralisant le mieux possible, c'est à dire le prédicteur qui permet de retrouver de manière la meilleure sortie y associée à une entrée x^i. Formellement, l'objectif est de retrouver la fonction f^* définie par "$f^* = \text{argmin}_{f\in\mathcal{F}} \int l(y, f(x))dP(x, y)$" où $\mathcal{F}$ est l'ensemble des fonctions considérées, l est une fonction de coût correspondant au prix à payer lorsque l'on labellise y par $f(x)$ et P est la distribution de probabilité ayant généré les données [1]. Généralement, les fonctions considérées diffèrent les unes des autres par

1. Pour éviter des problèmes de surapprentissage, il est courant d'ajouter à cette fonction de coût un terme de régularisation.

le biais d'un vecteur de k paramètres θ. L'objectif est alors de retrouver le meilleur vecteur de paramètres $\theta^* = \text{argmin}_{\theta \in \mathbb{R}^k} \int l(y, f_\theta(x)) dP(x, y)$. Dans les deux méthodes que nous allons considérer, la régression linéaire et les réseaux de neurones, le vecteur de paramètres correspond aux coefficients directeurs de l'hyperplan séparateur (ou interpolateur dans le cas d'une régression) dans le premier cas et aux poids synaptiques du réseau de neurones dans le second.

2.1.2 Régression linéaire

La régression linéaire est l'une des plus anciennes méthodes de régression. Elle est cependant toujours utilisée de nos jours en raison de sa simplicité et de son efficacité. Si l'on suppose que le vecteur d'entrée $\mathbf{x}$ peut s'écrire sous la forme $\mathbf{x} = (x_1, x_2, ..., x_k)$, une régression linéaire fournira une sortie y pouvant s'écrire sous la forme :

$$y = w_0 + \sum_{i=1}^{k} w_i x_i$$

Ici, le vecteur de paramètres $\mathbf{w} = (w_0, ..., w_N)$ est le vecteur à déterminer. Dans le cas le plus classique, en se donnant un jeu de données $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$, ces paramètres sont choisis de manière à minimiser la perte quadratique suivante :

$$l(\mathbf{w}) = \sum_{i=1}^{N} \Big[y_i - (w_0 + \sum_{j=1}^{k} w_j x_j) \Big]^2$$

Le vecteur de paramètres optimal, appelé solution optimale au sens des *moindres carrés*, peut s'écrire sous la forme suivante :

$$\mathbf{w}^* = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T y$$

2.1.3 Réseaux de neurones

Le principal inconvénient de la méthode exposée au paragraphe précédent est que par nature, elle ne peut séparer l'espace qu'en deux sous-espaces séparés par un hyperplan. Malheureusement, dans la majorité des problèmes d'apprentissage statistique, les données à labelliser ne sont pas linéairement séparables présentées sous forme brute. Pour pallier ce problème, une méthode couramment utilisée est, par le biais de transformations non-linéaires, de projeter ces données dans un nouvel espace où les méthodes de régressions linéaires sont applicables.

Cette opération peut cependant requérir une quantité considérable d'expertise humaine, nécessairement spécifique au problème. Les architectures d'apprentissage profond, de type réseaux de neurones, permettent d'automatiser cette étape d'extraction de caractéristiques discriminantes. L'idée commune à ces architectures est d'empiler (*i.e.* de composer) un certain nombre de classifieurs non-linéaires. L'ensemble des méthodes basées sur ce principe est appelé *apprentissage profond* (Goodfellow et al. (2016)). Cette méthode présente l'intérêt d'apprendre différents niveaux de représentations,

afin d'obtenir une représentation finale simplifiant la tâche de classification. À l'heure actuelle, dans la plupart des tâches d'apprentissage supervisé classiques (*i.e.* apprendre à transformer un vecteur en un autre vecteur), l'apprentissage profond domine les bancs d'essai.

2.1.3.1 Perceptron multi-couches

Le perceptron multi-couches (Rosenblatt (1958)) est une architecture simple de réseau de neurones à la base de tous les modèles d'apprentissage profond. Cette architecture est dite à propagation avant (ou *feedforward* en anglais) car elle ne comporte pas de cycles. Dans ce modèle d'architecture, l'information se propage uniquement dans le sens de l'entrée à la sortie à travers un ensemble de structures appelées couches. Dans ce modèle, le nombre de couches est appelé *profondeur* du réseau de neurones.

Formellement, si $f^{(k)}$ est la fonction associée à la couche k, la sortie $\mathbf{y}$ du réseau de neurones à N couches associée à un vecteur d'entrée $\mathbf{x}$ est donnée par $\mathbf{y} = f^{(N)}(f^{(N-1)}(...(f^{(1)}(\mathbf{x})...))$. Généralement, la fonction $f^{(k)}$, appelée fonction d'activation est composée d'une transformation linéaire (ou affine) $g_1^{(k)} : \mathbf{x} \mapsto W^{(k)}\mathbf{x} + b^{(k)}$ suivie d'une transformation non linéaire g_2 (telle une sigmoïde ou alors la fonction ReLu définie par $x \mapsto \max\{0, x\}$). Pour normaliser la sortie du réseau de neurones dans le cadre d'une tâche de classification, une dernière couche appelée *couche softmax* est généralement rajoutée au modèle (le score affecté à chaque catégorie correspond alors à la probabilité estimée que l'entrée corresponde à cette catégorie).

2.1.3.2 Réseaux de neurones récurrents

Les réseaux de neurones dits *feedforward* ne sont pas adaptés au traitement des séries temporelles. Si l'on se place dans le paradigme entrée-sortie, les entrées (ainsi que les sorties) doivent chacune former un unique bloc. Prenons l'exemple d'un problème de classification des énoncés d'un dialogue. Pour réaliser cette classification à l'aide d'un réseau de neurones de type *feedforward*, deux solutions sont possibles. On peut tout d'abord considérer chaque énoncé de manière indépendante et réaliser cette tâche de classification sur chacun de ces énoncés. Cette solution n'est cependant pas satisfaisante car dans de nombreux cas, des énoncés peuvent être classifiés de différentes manières et seul le contexte permet une désambiguïsation. La deuxième solution consiste à intégrer l'ensemble du contexte dans l'entrée. L'une des difficultés de cette solution est que les réseaux de neurones de type *feedforward* sont uniquement capables de traiter des vecteurs de taille fixe. Une solution possible serait de créer un vecteur d'entrée de la taille de la plus grande séquence possible. Cependant, cette taille n'est pas toujours définie et si elle se trouve trop grande, des limitations en temps de calcul et en mémoire apparaissent.

Les réseaux de neurones récurrents (en anglais *Recurrent Neural Networks*, RNN) apportent un moyen de traiter ce problème en introduisant des boucles de rétroaction à travers le réseau. Ainsi, au lieu d'associer simplement une entrée à une sortie, les réseaux récurrents associent à l'entrée une représentation compressée de l'historique (mise sous

forme vectorielle) et fournissent une sortie ainsi qu'une représentation compressée du nouvel historique : $(\mathbf{x}_t, \mathbf{h}_t) \mapsto (\mathbf{y}_t, \mathbf{h}_{t+1})$. Au cours du temps, l'information se propage donc dans le RNN comme présenté sur la Figure 2.1. Ainsi, si l'on reprend la tâche de classification exposée au paragraphe précédent, le premier énoncé du dialogue sera classifié par un réseau de neurones récurrent en lui associant l'historique nul. Le second sera classifié en lui associant le vecteur historique h_1 récupéré en sortie de la première classification, et ainsi de suite.

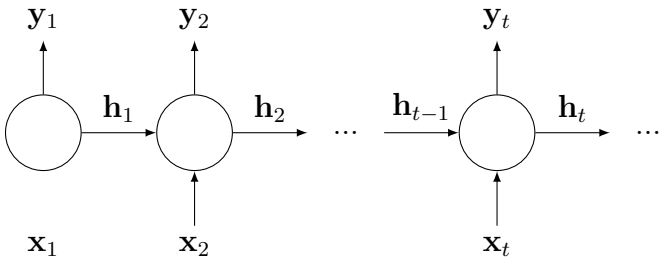


FIGURE 2.1 – Propagation de l'information à travers un RNN

LSTM Bien que les réseaux de neurones récurrents tels que décrits précédemment soient en théorie à même de représenter n'importe quel type de série temporelle, en pratique, il est difficile pour un réseau de ce type de ne garder que l'information utile dans le passé. Généralement, le passé lointain est oublié au profit du passé court terme.

Les LSTM (pour *Long Short Term Memory*) sont une architecture de réseau de neurones récurrents capables de gérer ces dépendances temporelles à long terme (Hochreiter and Schmidhuber (1997)). L'architecture d'une LSTM est représentée en Figure 2.2. Cette cellule est constituée de différents blocs, appelés état de la cellule, sortie et portes. Chaque bloc a une fonction précise. L'état de la cellule $\mathbf{C}_t$ est le composant principal de la LSTM. Son objectif est de représenter le passé de la manière la plus pertinente possible. Seules les modifications linéaires sont effectuées sur cet état de la cellule. Ces modifications peuvent être une suppression des informations passées non pertinentes, qui sont contrôlées par la porte d'oubli $\mathbf{f}_t$. Les informations présentes pertinentes peuvent quant à elles également être ajoutées à l'état de la cellule grace à la porte d'entrée $\mathbf{i}_t$. La sortie $\mathbf{h}_t$ est une forme filtrée de l'état de la cellule, qui est transformée afin de répondre aux objectifs du réseau de neurones.

Une LSTM s'implémente par les équations suivantes :

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(W_i c_t + U_i h_{t-1}) \\
\mathbf{f}_t &= \sigma(W_f c_t + U_f h_{t-1}) \\
\mathbf{C}_t &= i_t * \tanh(W_c c_t + U_c \mathbf{h}_{t-1}) + f_t * C_{t-1} \\
\mathbf{o}_t &= \sigma(W_o x_t + U_o h_{t-1}) \\
\mathbf{h}_t &= o_t * \tanh(C_t)
\end{aligned}
$$

avec $\mathbf{i}_t$ la porte d'entrée, σ la fonction sigmoïde, $\mathbf{f}_t$ la porte d'oubli, $\mathbf{o}_t$ la porte de sortie, $\mathbf{C}_t$ l'état de la cellule et $\mathbf{h}_t$ l'état caché.

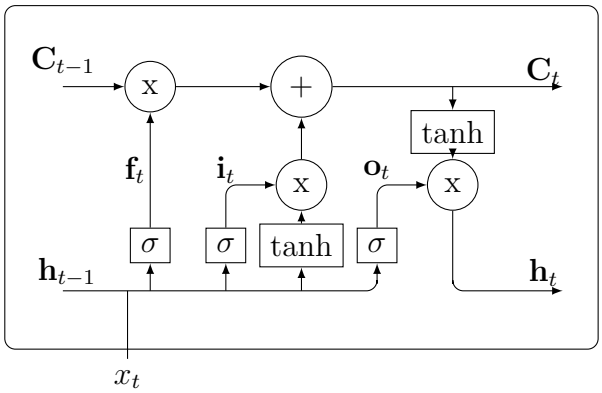


FIGURE 2.2 – LSTM

Sequence to Sequence Les réseaux de neurones récurrents présentés précédemment associent un vecteur de sortie à une séquence temporelle. Les modèles de type *Sequence to Sequence* (ou *Seq2Seq*) associent, comme leur nom indique, des séquences temporelles à des séquences temporelles (pas nécessairement de mêmes tailles). Ils sont très utilisés dans le domaine du dialogue.

Ces modèles fonctionnent comme représentés en Figure 2.3. Tout d'abord un réseau de neurones récurrents, appelé *encodeur*, produit un vecteur (ou une séquence de vecteurs) appelé contexte C représentant de manière compressée la séquence d'entrée $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n_x})$. Prenant en entrée ce contexte, un second réseau de neurones récurrents, appelé *décodeur*, génère la séquence de sortie $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_{n_y})$. Ces deux réseaux de neurones sont entraînés de manière à maximiser la probabilité $P(\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_{n_y} | \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n_x})$ sur l'ensemble d'entraînement.

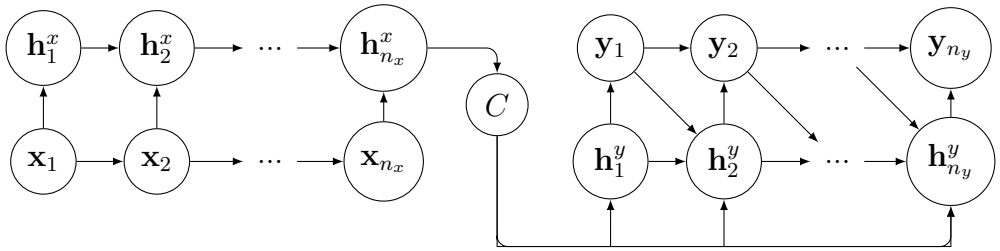


FIGURE 2.3 – Architecture *Seq2Seq*

2.1.3.3 Entraînement des réseaux de neurones

En raison de la structure particulière de la fonction de coût associée aux réseaux de neurones, ceux-ci sont généralement entraînés par des méthodes itératives de type

descente de gradient (et non pas par des méthodes d'algèbre linéaire comme dans le cas des méthodes linéaires).

La descente de gradient est une méthode d'optimisation exploitant le fait que, en modifiant les paramètres θ du réseau de neurones dans la direction de la "pente" de la fonction de coût C, on convergera vers un minimum local de la fonction C. Formellement, les paramètres du réseau de neurones sont modifiés grâce à l'équation de mise à jour $\theta \leftarrow C - \alpha\frac{\partial C}{\partial \theta}$ où α est un paramètre appelé pas d'apprentissage (en anglais, *learning rate*). Ce gradient est calculé grâce à l'algorithme de *rétropropagation du gradient*, essentiellement basé sur la formule de dérivation en chaîne.

Généralement, ce gradient n'est pas calculé sur l'ensemble des données d'entraînement mais sur un mini-lot d'échantillons (appelé *mini batch*). Cette implémentation est appelée *descente du gradient stochastique* (en anglais *Stochastic Gradient Descent*, SGD). Elle est préférée à la descente de gradient classique pour des raisons de rapidité et de convergence.

En raison du caractère non convexe de la fonction de coût, il n'existe pas de garantie de convergence vers le minimum global de la fonction de coût pour cet algorithme. En pratique cependant, la convergence vers un minimum local (ou vers un point selle) est suffisante pour obtenir de très bons résultats.

2.2 Prise de décision séquentielle

L'apprentissage par renforcement (*Reinforcement Learning*, RL) est le domaine de l'apprentissage automatique s'intéressant à l'optimisation à partir de données d'un comportement dans un environnement incertain. Le principe fondamental de l'apprentissage par renforcement est d'adapter un comportement en le confrontant à la pratique en situation réelle, *i.e.* d'apprendre par essais-erreurs. Un système apprenant par renforcement cherche donc à répondre à la question suivante : comment me comporter de manière à maximiser sur le long terme la somme de mes récompenses futures ? Quel est le bon compromis entre une petite récompense instantanée maintenant et une grosse récompense incertaine plus tard ?

Historiquement, c'est dans la psychologie animale où l'apprentissage de comportements par essais-erreurs a été étudié pour la première fois (Thorndike (1898), Skinner (1938)). Ce dernier s'intéressait à l'apprentissage des animaux par le conditionnement et nomma renforcement le processus conduisant l'animal à adopter des comportements produisant un stimulus positif et évitant ceux amenant un stimulus négatif.

Ce n'est que bien plus tard, avec le développement de la programmation dynamique dans Bellman (1954) que l'utilisation d'une fonction de récompense fut utilisée pour la première fois dans la conception de systèmes. Contrairement à l'apprentissage, cette théorie ne se base pas sur des données. Les paramètres du système à optimiser sont supposés connus et le *contrôle optimal*, *i.e.* la séquence de décisions optimale, est ensuite calculé via des algorithmes adaptés à ce cadre.

En dépit de quelques rares exceptions, il faudra attendre Klopf (1972), Barto and

Sutton (1981) pour voir émerger une vraie théorie de l'optimisation de systèmes par essais-erreurs (*i.e.* par apprentissage à partir de données).

Dès lors, l'apprentissage par renforcement a connu de nombreux succès que ce soit les systèmes de dialogue (Levin and Pieraccini (1997), Singh et al. (2000)), le pilotage d'un hélicoptère (Ng et al. (2006)), les jeux de plateau tels le backgammon (Tesauro (1994)) ou le jeu de go (Silver et al. (2016)), les jeux vidéos ATARI (Mnih et al. (2015)) dont Ms. Pac-Man (Van Seijen et al. (2017)) ou Montezuma Revenge (Hester et al. (2017a), OpenAI (2018b)), DOTA (OpenAI (2018a)), la réduction de consommation énergétique (Evans and Gao (2016)), *etc.*

2.2.1 Programmation dynamique

La programmation dynamique est une formalisation de la problématique de décision séquentielle dans un environnement incertain. Dans ce cadre, on considère un agent prenant des décisions à un pas de temps régulier. Chacune de ces décisions impacte l'environnement dans lequel il se trouve. L'objectif de l'agent est alors de choisir la bonne séquence d'actions afin d'accomplir ses objectifs. De manière générale, l'objectif de la programmation dynamique est de définir des algorithmes permettant de trouver cette bonne séquence quelque soit l'environnement.

Les Processus Décisionnels de Markov (*Markov Decision Process*, MDP) sont des modèles stochastiques permettant d'aborder de manière formelle le problème de la programmation dynamique (Puterman (2014)). Ils sont définis de la façon suivante :

Définition 2.1. *Un Processus Décisionnel de Markov est un quintuplet* $\{\mathcal{S}, \mathcal{A}, R, T, \gamma\}$ *où :*

- $\mathcal{S}$ *est l'espace d'état*
- $\mathcal{A}$ *est l'espace d'actions*
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *est la fonction de récompense (possiblement stochastique)*
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *est la fonction de transition*
- $\gamma \in [0, 1[$ *est un facteur d'amortissement*

Accomplir l'action $a \in \mathcal{A}$ dans un état $s \in \mathcal{S}$ mène à un état s' échantillonné selon la fonction de transition $T(s, a, .)$ ($T(s, a, s')$ correspond à la probabilité de se retrouver dans l'état s' après avoir executé) et à une récompense r échantillonnée selon la fonction de récompense $R(s, a)$. Une politique π est une fonction (possiblement stochastique) $\pi : \mathcal{S} \to \mathcal{A}$. L'objectif de la programmation dynamique est de trouver l'une des politiques maximisant l'espérance de la somme amortie des récompenses futures $\mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_k))\Big]$.

Un MDP dépeint donc de manière formelle un agent évoluant dans un monde et agissant sur celui-ci. Selon ce paradigme, nous considérons un agent agissant de manière synchrone. A chaque instant t, l'agent accomplit une action $a = \pi(s)$ dans l'état s, cette action est définie par sa politique. Ayant accomplit cette action, son environnement le place dans un nouvel état s' et lui donne une récompense r ainsi que l'indique la Figure 2.4.

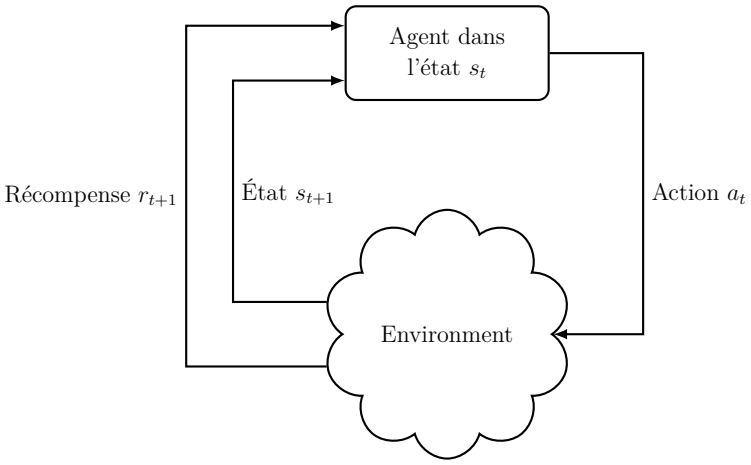


FIGURE 2.4 – Processus décisionnel de Markov

2.2.2 Fonctions de valeur

Il est possible d'associer une valeur à la politique d'un agent. On définit celle-ci comme l'espérance de la somme amortie des récompenses d'un agent lorsque, en commençant dans un certain état s, il suit indéfiniment cette politique. Formellement, cette valeur s'écrit :

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \Big| s_0 = s\Big].$$

À chaque pas de temps t, l'agent cherche à maximiser la somme espérée de ses futures récompenses, définie par $V(s_t) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, \pi(s_k))\Big]$. Cette quantité est appelée *fonction de valeur*. En la reformulant, nous obtenons :

$$\begin{aligned} V^{\pi}(s_t) &= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_k))\Big] \\ &= \mathbb{E}\Big[R(s_t, \pi(s_t))\Big] + \mathbb{E}\Big[\sum_{k=1}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k}))\Big] \\ &= r(s_t, \pi(s_t)) + \gamma \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k}))\Big] \\ &= r(s_t, \pi(s_t)) + \gamma \sum_{s' \in \mathcal{S}} T(s_t, \pi(s_t), s') V(s') \end{aligned}$$

Nous pouvons alors observer que le facteur d'amortissement γ, introduit pour garantir l'existence de V, pousse l'agent à préférer une récompense immédiate à la même récompense plus tard, traduisant une forme d'*impatience*.

On dit qu'une fonction de valeur V est supérieure (respectivement inférieure) à une autre fonction de valeur V' si dans chaque état $s \in \mathcal{S}$, $V(s) \geq V'(s)$ (respectivement $V(s) \leq V(s')$). On peut montrer que dans chaque MDP, il existe une unique fonction de valeur supérieure à toutes les autres. Cette fonction est nommée *fonction de valeur*

optimale et est notée V^*. L'objectif est donc de retrouver une politique π^*, appelée politique optimale et non nécessairement unique, associée à cette fonction de valeur.

Doté de la fonction de valeur optimale, il n'est pas nécessairement facile de retrouver une politique associée. La Q-fonction est une fonction semblable à la fonction de valeur permettant de retrouver facilement cette politique. Elle est définie comme la fonction de valeur associée à une politique, moyennant un relâchement de la première action. Formellement, elle est définie par :

$$\begin{aligned}\forall (s,a) \in \mathcal{S} \times \mathcal{A},, Q(s,a) &= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V^{\pi}(s') \\ &= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') Q^{\pi}(s', \pi(s'))\end{aligned}$$

On note Q^* la Q-fonction associée à la politique optimale. Connaissant Q^*, on peut dériver une politique optimale par $\forall s \in \mathcal{S}$, $\pi^*(s) \in \text{argmax}_{a \in \mathcal{A}} Q(s,a)$.

2.2.3 Algorithmes de résolution exacte

2.2.3.1 Itération de la valeur

L'algorithme d'*itération de la valeur* est une procédure itérative utilisant l'équation de Bellman comme règle de mise à jour. Cette approche, de type point fixe, tire partie de la convergence de la suite définie par $V_{n+1} = LV_n$ (où L est l'opérateur dit *opérateur de Bellman* défini par $\forall s \in \mathcal{S}$, $L : V \mapsto \max_a \{r(s,a) + \gamma \sum_{s'} P(s,a,s') V_n(s')\}$ vers la fonction de valeur optimale, quelle que soit l'initialisation.

Comme il n'est en pratique pas possible de répéter la procédure sur une infinité d'itérations, la procédure d'arrêt la plus classique stoppe l'itération lorsque $||V_{n+1} - V_n|| < \epsilon$, où ϵ est un seuil fixé a priori.

L'algorithme d'itération de la valeur est défini par l'algorithme 1.

Algorithme 1 Algorithme d'itération de la valeur

Initialisation : Initialiser $n = 0$, V_n
Tant que $||V_{n+1} - V_n|| < \epsilon$ **faire**
  Pour $s \in \mathcal{S}$ **faire**
    $V_{n+1}(s) = \max_a \{r(s,a) + \gamma \sum_{s'} P(s,a,s') V_n(s')\}$
  Fin pour
  $n \leftarrow n + 1$
Fin tant que
Pour $s \in \mathcal{S}$ **faire**
  $\pi(s) \in \text{argmax}_a \{r(s,a) + \gamma \sum_{s'} P(s,a,s') V_n(s')\}$
Fin pour
Retourner V_n, π

2.2.3.2 Itération de la politique

L'algorithme d'itération de la valeur présente l'inconvénient de requérir une infinité d'itérations pour converger vers la fonction de valeur optimale. Il existe cependant à cet algorithme une alternative appelée *itération de la politique* convergeant toujours en un temps fini.

L'algorithme d'itération de la politique repose sur le résultat suivant :

Théorème 2.2.1. *Soit une politique π et sa fonction de valeur V^π. La politique π' définie par*

$$\forall s \in \mathcal{S},\ \pi'(s) \in \operatorname*{argmax}_a \{r(s,a) + \gamma \sum_{s'} P(s,a,s')V^\pi(s')\},$$

gloutonne vis à vis de V^π, résultera en une fonction de valeur $V^{\pi'}$ telle que $V^{\pi'} \geq V^\pi$ avec $V^{\pi'} = V^\pi$ si et seulement si V^π est la fonction de valeur optimale V^*.

Ce théorème est également valide pour les Q-fonctions, qui permettent de dériver immédiatement une politique gloutonne.

Chaque itération de la procédure d'itération de la politique se divise en deux étapes. Dans un premier temps, lors d'une étape appelée *évaluation de la politique*, on cherche la Q-fonction Q^π associée à la politique π_n en résolvant l'équation : $\forall s \in \mathcal{S},\ Q^{\pi_n}(s,a) = r(s,a)) + \gamma \sum_{s' \in \mathcal{S}} P(s,a,s')Q^{\pi_n}(s', \pi_n(s'))$. De manière compacte, cette équation se réécrit : $Q^{\pi_n} = L^{\pi_n} Q^{\pi_n}$ où L^{π_n} est l'opérateur de Bellman associé à la politique π_n. La deuxième étape, appelée *amélioration de la politique*, cherche à retrouver la politique gloutonne vis à vis de la fonction de valeur calculée lors de l'étape précédente : $\pi_{n+1}(s) \in \operatorname{argmax}_a \{r(s,a) + \gamma \sum_{s'} P(s,a,s')Q^{\pi_n}(s', \pi_n(s'))\}$.

L'algorithme d'itération de la politique s'obtient alors en recommençant ces deux opérations jusqu'à ce que $\pi_{n+1} = \pi_n$:

Algorithme 2 Algorithme d'itération de la politique

Initialisation : Initialiser $n = 0$, π_0
Tant que $\pi_{n+1} \neq \pi_n$ **faire**
  Résoudre $\forall s \in \mathcal{S},\ Q^{\pi_n}(s,a) = r(s,a)) + \gamma \sum_{s' \in \mathcal{S}} P(s,a,s')Q^{\pi_n}(s', \pi_n(s'))$
  Pour $s \in \mathcal{S}$ **faire**
    $\pi_{n+1}(s) \in \operatorname{argmax}_a \{r(s,a) + \gamma \sum_{s'} P(s,a,s')Q^{\pi_n}(s', \pi_n(s'))\}$
  Fin pour
  $n \leftarrow n + 1$
Fin tant que
Retourner π_n

2.2.4 Algorithmes de résolution approchée

Les deux algorithmes précédents supposent qu'il est possible de stocker pour chaque état sa valeur, ce qui peut s'avérer impossible lorsque le nombre d'états est grand, voir infini.

Pour traiter ce problème, l'approche la plus courante, appelée *approximation de la fonction de valeur*, est d'extraire de chaque état s un vecteur de N caractéristiques $\Phi(s) = \{\phi_i(s)\}_{i=1}^N$ (appelées *features*). De ces composantes est alors régressée la fonction de valeur à l'aide des méthodes exposées dans la section précédente, menant alors à des algorithmes de résolution approchée. La fonction de valeur V n'est alors plus représentée comme un tableau mais comme une fonction paramétrée par un vecteur θ. Ainsi, la fonction de valeur associée à chaque état s est $V_\theta(s) = f_\theta(\Phi(s))$ où f_θ est un prédicteur et où, par souci de légèreté dans les notations, l'état est confondu avec ses composantes.

2.2.4.1 Algorithme d'itération de la valeur avec approximation

L'intuition derrière les algorithmes de programmation dynamique approchée est donnée par le résultat suivant : si la fonction de valeur approchée V_θ est proche de la fonction de valeur optimale, alors la valeur de la politique gloutonne par rapport à la fonction de valeur approchée sera proche de la fonction de valeur optimale. Ainsi, on espère que l'erreur commise en considérant, à chaque itération des algorithmes de programmation dynamique, la politique gloutonne par rapport à la fonction de valeur estimée au lieu de la vraie fonction de valeur, est suffisamment faible pour ne pas impacter la convergence des algorithmes. On ne peut malheureusement pas garantir dans le cas général une convergence de ces algorithmes.

Comme indiqué en Figure 2.5, l'algorithme d'itération de la valeur (IVA) avec approximation fonctionne de manière similaire à l'algorithme d'itération de la valeur. Cependant, à chaque itération de cet algorithme, un ensemble fini d'états est tiré aléatoirement. La fonction de valeur de chacun de ces états est ensuite calculée à l'aide de l'équation de Bellman. Puis cette fonction est régressée sur l'ensemble des états $\mathcal{S}$.

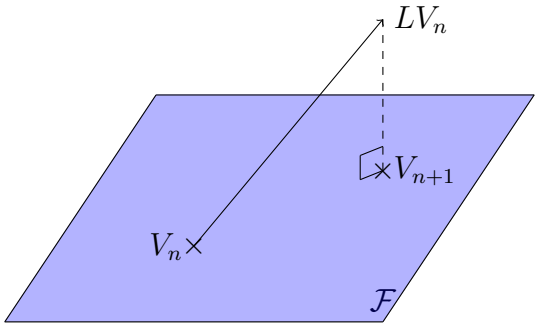


FIGURE 2.5 – Représentation schématique d'une itération de l'algorithme d'itération de la valeur avec approximation : À chaque étape de la procédure, l'opérateur de Bellman optimal est appliqué à la fonction de valeur V_n, puis est régressé sur l'espace d'hypothèse $\mathcal{F}$ pour donner V_{n+1}

L'algorithme d'itération de la valeur avec approximation est donc l'Algorithme 3.

Algorithme 3 Algorithme d'itération de la valeur avec approximation

Initialisation : Initialiser $n = 0$, V_n
Tant que $||V_{n+1} - V_n|| < \epsilon$ **faire**
  Choisir $(s_k)_{k=1}^K \subset \mathcal{S}$
  Pour $k = 1..K$ **faire**
    $v_k = \max_a\{r(s,a) + \gamma \sum_{s'} P(s,a,s')V_n(s')\}$
  Fin pour
  $V_{n+1} = f((s_k, v_k)_{k=1}^N)$ où f est un prédicteur d'apprentissage supervisé
  $n \leftarrow n + 1$
Fin tant que
Pour $s \in \mathcal{S}$ **faire**
  $\pi(s) \in \text{argmax}_a\{r(s,a) + \gamma \sum_{s'} P(s,a,s')V_n(s')\}$
Fin pour
Retourner V_n, π

2.2.4.2 Algorithme d'itération de la politique avec approximation

Comme IVA, l'algorithme d'itération de la politique avec approximation (IPA) est très similaire à l'algorithme d'itération de la politique. Cependant, durant la phase d'évaluation de la politique, la fonction de valeur retenue sera une fonction de valeur obtenue par régression.

Algorithme 4 Algorithme d'itération de la politique avec approximation

Initialisation : Initialiser $n = 0$, π_0
Tant que $\pi_{n+1} \neq \pi_n$ **faire**
  Calculer une approximation Q_n de la fonction de valeur Q^{π_n}
  Pour $s \in \mathcal{S}$ **faire**
    $\pi(s) \in \text{argmax}_a\{r(s,a) + \gamma \sum_{s'} P(s,a,s')Q_n(s', \pi_n(s'))\}$
  Fin pour
  $n \leftarrow n + 1$
Fin tant que
Retourner π_n

Afin d'évaluer correctement la politique, dans le cas classique d'une approximation linéaire, il est très courant d'utiliser des méthodes de type moindres carrés. Dans ce cadre, en représentant chaque couple état-action par un vecteur de coefficients $(s,a) := \phi(s,a) = (\phi(s,a)_1 ... \phi(s,a)_d)$, la Q-fonction recherchée doit pouvoir s'écrire sous la forme du produit scalaire $\hat{Q} = \langle s, \theta \rangle$ où θ est un vecteur de paramètres à déterminer. Représenter la Q-fonction de cette manière permet alors d'évaluer la politique selon deux approches différentes, une de type *minimisation du résidu de Bellman* et une de type *minimisation de l'erreur des différences temporelles*, dont la différence est représentée en Figure 2.6. Dans les deux cas, retrouver la fonction de valeur revient à résoudre un système d'équations linéaires que nous allons développer.

Introduisons tout d'abord les matrices suivantes :

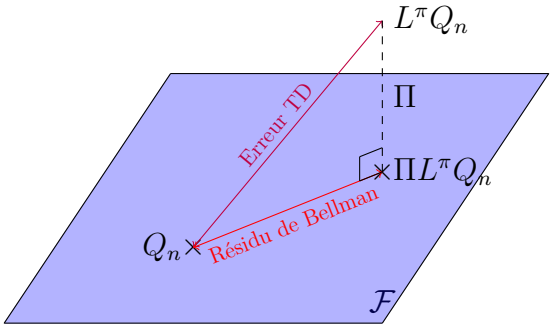


FIGURE 2.6 – Représentation schématique de la différence entre l'erreur des différences temporelles (erreur TD, *en violet*) et le résidu de Bellman (*en rouge*)

- $\underline{Q} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ est la représentation vectorielle de la Q-fonction
- $\underline{R} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}||}$ est la représentation vectorielle de l'espérance de la fonction de récompense
- $\underline{T^\pi} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ est la représentation vectorielle de la fonction de transition associée à une politique π. $\underline{T^\pi}_{((i,j),(i',j'))} = T(s_i, a_j, s_{i'})$ si $\pi(s_{i'}) = a_{j'}$ et 0 sinon
- $\underline{\Phi} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times d}$ est la représentation matricielle de la représentation vectorielle des états
- $\underline{\rho} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ est une matrice de poids définissant la probabilité de distribution des couples états-actions

Dans le cas de la minimisation de l'erreur des différences temporelles, on recherche une Q-fonction minimisant la quantité $\left|\left|\hat{Q}^\pi(s,a) - \Pi\left[r(s,a) + \gamma \sum_{s'} P(s,a,s')\hat{Q}^\pi(s,a)\right]\right|\right|$, où Π représente l'opérateur de projection sur l'ensemble des Q-fonctions représentables. Il peut être prouvé que dans un tel cas le vecteur θ peut être trouvé en résolvant l'équation :

$$(A - \gamma B)\theta = b$$

avec $A = \underline{\Phi}^T \underline{\rho} \Phi$, $B = \underline{\Phi}^T \underline{\rho} T^\pi \underline{\Phi}$ et $b = \underline{\Phi}^T \underline{\rho} R$.

Dans le cas de la minimisation directe du résidu de Bellman, on cherche à résoudre le problème d'optimisation $\min_{\hat{Q}\in\hat{\mathcal{Q}}} \left|\left|\hat{Q}^\pi(s,a) - \left[r(s,a) + \gamma \sum_{s'} P(s,a,s')\hat{Q}^\pi(s,a)\right|\right|\right|$ où $\hat{\mathcal{Q}}$ est l'ensemble des Q-fonctions représentables linéairement. Dans ce cas, la résolution du problème se fait alors en résolvant l'équation :

$$C^T \underline{\rho} C\theta = C^T \underline{\rho} R$$

où $C = \underline{\Phi} - \gamma \underline{T^\pi} \underline{\Phi}$.

2.2.5 Apprentissage par renforcement

L'apprentissage par renforcement (*Reinforcement Learning*, RL) se base essentiellement sur la théorie de la programmation dynamique. Cependant, une hypothèse de celle-ci est relachée. En effet, les algorithmes précédents supposent connues les fonctions

de transition et de récompense, ce qui n'est que très rarement le cas en pratique. Ainsi, un agent apprenant par renforcement ne peut apprendre ces fonctions que localement par l'expérimentation.

Un agent apprenant par renforcement apprend à partir de données. De ces données, soit il se construira explicitement une représentation de son environnement en cherchant à retrouver le plus précisément possible les fonctions de récompense et de transition, soit de manière implicite il cherchera uniquement suffisamment d'informations pour prendre les meilleures décisions possibles.

Ces données peuvent soit provenir d'un corpus d'interactions préalablement récolté, soit de l'interaction immédiate avec l'environnement. Les méthodes d'apprentissage diffèrent selon que les données parviennent de la première ou deuxième manière. Dans le premier cas, on parlera d'apprentissage hors-ligne, ou *batch* (*off-line*) tandis que dans le second cas, on parle d'apprentissage en ligne (*on-line*).

2.2.5.1 Approches en ligne

L'apprentissage par renforcement en ligne est l'apprentissage par essais-erreurs. À chaque pas de temps t, l'agent, ayant exécuté l'action a_t dans l'état s_t reçoit une récompense r_t et met à jour sa politique π_t en fonction de cette récompense.

Q-learning L'algorithme Q-Learning (Watkins and Dayan (1992)) est l'un des algorithmes les plus anciens et les plus utilisés de l'apprentissage par renforcement. Un agent apprenant par Q-learning cherche à approximer la Q-fonction en agissant dans son environnement et en mettant à jour sa propre Q-fonction estimée. Quelle que soit l'initialisation de l'algorithme, la mise à jour suivante : $Q(s_t, a_t) \leftarrow S(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big]$ assure une convergence de la Q-fonction vers la Q-fonction optimale sous réserve de visiter chaque couple état-action une infinité de fois et moyennant de respecter certaines hypothèses sur la décroissance du paramètre α. Q-learning peut se décrire par l'Algorithme 5

Algorithme 5 Q-Learning

Entrées : Une Q-fonction initiale quelconque, un paramètre d'apprentissage α
Pour Chaque épisode **faire**
  Tant que L'épisode n'est pas terminé **faire**
    Choisir une action $a \in \mathcal{A}$ à exécuter
    Récupérer l'état suivant s' et la récompense reçue $r(s, a)$
    Mettre à jour $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big]$
  Fin tant que
Fin pour

Dilemme exploration-exploitation Quel que soit l'algorithme d'apprentissage par renforcement utilisé, il est toujours nécessaire de disposer d'un certain nombre de données sur l'environnement afin de mieux approximer celui-ci en vue de toujours prendre

la meilleure décision possible. Un compromis dans le choix des actions se pose alors : vaut-il mieux exploiter ce que je connais ou vaut il mieux prendre des décisions me permettant d'explorer mon environnement afin de prendre de meilleures décisions plus tard ?

Ce problème est appelé *dilemme exploration-exploitation*. Une manière simple et efficace de le régler se nomme la stratégie ϵ-gloutonne. Dans chaque état, avec une probabilité ϵ, l'agent choisit de prendre une action aléatoirement. Le reste du temps, l'agent choisit l'action correspondant à la meilleure Q-fonction. Bien que cette stratégie soit très simple, c'est à l'heure actuelle l'une des plus efficace.

2.2.5.2 Approches hors ligne

Les méthodes d'apprentissage par renforcement hors-ligne (aussi appelé apprentissage par renforcement *batch*) n'apprennent pas par interactions directes avec l'environnement mais à partir d'un ensemble de données préalablement collectées, sous la forme d'un corpus.

Former un corpus est un exercice délicat. Comme abordé en introduction, le dialogue humain/humain et le dialogue humain/machine n'obéissent pas tout à fait aux mêmes règles. Ainsi, un corpus de dialogues humain/machine se doit d'être collecté, ce qui peut être problématique si l'on ne dispose pas préalablement d'un système de dialogue. Une méthode populaire pour pallier ce problème est celle du Magicien d'Oz où un humain dialogue avec un opérateur se faisant passer pour une machine (Edlund et al. (2008)).

Un système apprenant hors-ligne n'apprend donc pas par l'interaction directe. Les algorithmes d'apprentissage par renforcement hors-ligne se basent principalement sur les algorithmes de programmation dynamique avec approximation présentés en Section 2.2.4, introduisant une approximation de la fonction de transition et de la fonction de récompense.

Les méthodes par renforcement hors-ligne supposent tout d'abord que l'on soit doté d'un ensemble d'échantillons $\mathcal{D} = \{s_i, a_i, s'_i, r_i\}_{i=1}^N$ obtenu par une certain politique d'exploration. Comme montré sur la Figure 2.7, l'objectif est alors de dériver une politique à partir de ces échantillons uniquement, puis d'appliquer cette politique telle quelle dans le monde réel.

FIGURE 2.7 – Apprentissage par renforcement hors-ligne

La taille de l'ensemble de données étant fini, il n'est généralement pas possible de dériver la politique optimale à partir de celui-ci. L'objectif est donc de retrouver la meilleure politique possible. Bien que conceptuellement, un tel objectif puisse sembler moins intéressant que celui de l'apprentissage par renforcement en ligne, ces méthodes

ont connu de nombreux succès de part leur stabilité et leur efficacité en données et en temps, notamment sur des problèmes dotés de grands espaces d'états et d'actions. Par ailleurs, dans de nombreuses applications industrielles, le principe de l'apprentissage par renforcement hors-ligne est conceptuellement intéressant. En effet dans ce cadre, une politique implémentée à base de règles peut d'abord être utilisée pour collecter des données de manière sûre, puis un algorithme d'apprentissage hors-ligne peut être utilisé pour dériver une meilleure politique à partir de ces données.

Comme pour les algorithmes de programmation dynamique, il existe deux familles d'algorithmes de renforcement hors ligne, l'un se basant sur une itération de la valeur et l'autre sur une itération de la politique.

Fitted-Q iteration Fitted-Q iteration (ou simplement Fitted-Q), exposé en Figure 6, est un algorithme d'apprentissage par renforcement hors-ligne se basant sur l'algorithme d'itération de la valeur avec approximation (Ernst et al. (2005), Riedmiller (2005)).

Pour utiliser Fitted-Q, on suppose que l'on est doté d'un ensemble d'échantillons $\mathcal{D} = \{s_i, a_i, s'_i, r_i\}_{i=1}^N$ et que la Q-fonction $\hat{Q}_0$ est initialisée (par exemple mise à zéro sur tous les couples état-action). L'ensemble $\mathcal{D}$ peut alors être étendu à $\overline{\mathcal{D}} = \{s_i, a_i, s'_i, r_i, q_{0,i}\}_{i=1}^N$, où q_i est la Q-valeur associée à l'échantillon i. L'algorithme répète ensuite les deux étapes suivantes. À l'itération k, $\overline{\mathcal{D}}$ est actualisé en actualisant les Q-valeurs en leur affectant $q_{k+1,i} = r_i + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_k(s'_i, a')$. Dans un second temps, c'est la Q-fonction $\hat{Q}$ qui est actualisée en la régressant sur l'ensemble $\overline{\mathcal{D}}$ (le nom de l'algorithme vient donc du fait que l'on cherche une Q-fonction *fittant* au mieux les données).

Dans les articles originaux, Ernst et al. (2005) propose d'utiliser des arbres de décision pour régresser la Q-fonction ; Riedmiller (2005) propose quant à lui d'utiliser des réseaux de neurones.

Algorithme 6 Fitted-Q

Entrées : Un ensemble de transitions $\mathcal{D} = \{s_i, a_i, s'_i, r_i\}_{i=1}^N$, $q_0 = 0$ une Q-fonction, $\mathcal{F}$ un espace d'hypothèses
Pour $k = 1..K$ **faire**
  Pour $i = 1..N$ **faire**
    Mettre à jour $q_{i,k} = r_i + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_k(s'_i, a')$
  Fin pour
  Résoudre le problème d'apprentissage supervisé $\hat{Q}_k = \operatorname{argmin}_{q \in \mathcal{F}} \sum_{i=1}^N l(q_{i,k}, q)$ où l est une fonction de coût
Fin pour
Retourner $\pi_K(s) \in \operatorname{argmax}_a \hat{Q}_K(s, a)$, $\forall s \in \mathcal{S}$

LSPI L'algorithme LSPI (pour *Least-Squares Policy Iteration*) est le pendant de Fitted-Q se basant sur l'algorithme d'itération de la politique avec approximation

(Lagoudakis and Parr (2003)). À l'instar de Fitted-Q, utiliser LSPI suppose que l'on soit doté d'un ensemble d'échantillons $\mathcal{D} = \{s_i, a_i, s'_i, r_i\}_{i=1}^N$. Cependant, LSPI impose le choix de la méthode d'approximation de la Q-fonction, qui doit être linéaire. Ainsi, en représentant chaque couple état-action par un vecteur de coefficients $(s,a) := \phi(s,a) = (\phi(s,a)_1 ... \phi(s,a)_M)$, la Q-fonction recherchée doit pouvoir s'écrire sous la forme du produit scalaire $\hat{Q} = \langle s, \theta \rangle$ où θ est un vecteur de paramètres à déterminer.

Comme l'algorithme d'itération de la politique, LSPI se décompose en deux étapes : une étape d'évaluation de la politique pouvant être résolue par la méthode des différences temporelles ou par minimisation du résidu de Bellman ; et une étape d'amélioration de la politique. La première étape donne lieu à une mise à jour du paramètre θ afin d'évaluer au mieux la politique courante.

Dans cas d'une évaluation de la politique avec la méthode des différences temporelles, on recherche comme dans la Section 4 à résoudre l'équation suivante :

$$(A - \gamma B)\theta = b.$$

Dans la mesure où les paramètres du MDP ne sont plus connus, les matrices A, B et le vecteur b sont construits à partir des échantillons suivant :

$$\begin{aligned} A_i &= A_{i-1} + \phi(s_i, a_i)\phi^T(s_i, a_i) \\ B_i &= B_{i-1} + \phi(s_i, a_i)\phi^T(s'_i, \pi(s'_i)) \\ b_i &= b_{i-1} + \phi(s_i, a_i) r_i \end{aligned}$$

Les méthodes cherchant à résoudre cette équation de manière directe sont dites de type LSTD (pour *Least-Squares Temporal Difference*), celles la résolvant de manière itérative sont dits de type LSPE (pour *Least-Squares Policy Evaluation*). L'algorithme LSPI complet en utilisant LSTD pour l'étape d'évaluation de la politique est développé à l'Algorithme 7

Dans le cas de la minimisation directe du résidu de Bellman, on résout l'équation $\widehat{C^T \rho C}\theta = \widehat{C^T \rho R}$ où les matrices $\widehat{C^T \rho C}$ et $\widehat{C^T \rho R}$ sont obtenues par :

$$\begin{aligned} \widehat{C^T \rho C}_i &= \widehat{C^T \rho C}_{i-1} + [\phi(s_i, a_i) - \gamma\phi(s'_{1,i}), \pi(s'_{1,i})][\phi^T(s_i, a_i) - \gamma\phi^T(s'_{2,i}), \pi(s'_{2,i})] \\ \widehat{C^T \rho R}_i &= \widehat{C^T \rho R}_{i-1} + \phi(s_i, a_i) - \gamma\phi^T(s'_{2,i}), \pi(s'_{2,i}) \end{aligned}$$

Algorithme 7 LSPI

Entrées : Un ensemble de transitions $\mathcal{D} = \{s_i, a_i, s'_i, r_i\}_{i=1}^N$, une politique π_0, un seuil ϵ, $j = 0$

Tant que $||Q^{\pi_k} - Q^{\pi_{k-1}}|| > \epsilon$ **faire**

$j = j + 1$

Évaluation de la politique

$\widehat{(A - \gamma B)}_0 = \underline{0}$, $\widehat{b_0} = \underline{0}$

Pour $i = 1..N$ **faire**

$\widehat{(A - \gamma B)}_i = \widehat{(A - \gamma B)}_{i-1} + \phi(s_i, a_i)\phi^T(s_i, a_i)_\gamma \phi(s_i, a_i)\phi^T(s'_i, \pi(s'_i))$

$\widehat{b_i} = \widehat{b_{i-1}} + \phi(s_i, a_i)r_i$

Fin pour

Résoudre $\frac{1}{N}\widehat{(A - \gamma B)}_N \theta = \frac{1}{N}\widehat{b_n}$

Amélioration de la politique

$\forall s \in \mathcal{S}$, $\pi_j \in \text{argmax}_{a \in \mathcal{A}} \phi^T(s, a)\theta$

Fin tant que

Retourner π_j

Chapitre 3
Systèmes de dialogue

Dans le chapitre précédent, nous avons présenté deux types d'apprentissages : l'apprentissage supervisé dont le rôle est d'apprendre à généraliser l'information extraite dans des situations connues à des situations inconnues et l'apprentissage par renforcement, dont l'objectif est d'apprendre à améliorer un comportement en vue d'optimiser la réalisation d'objectifs. Dans ce chapitre, après une présentation de l'architecture des systèmes de dialogue, nous allons voir de quelle manière ces deux types d'apprentissage sont utilisés pour la conception de tels systèmes.

3.1 Systèmes de dialogue

3.1.1 Une architecture modulaire

Un système de dialogue se conçoit généralement sous la forme d'une chaîne de cinq modules (Jurafsky and Martin (2009)). Dans l'ordre du traitement de l'information, ces modules sont les suivants : la reconnaissance vocale (*Automatic Speech Recognition*, ASR), la compréhension du langage parlé (*Spoken Language Understanding*, SLU), le gestionnaire de dialogue (*Dialogue Manager*, DM), la génération automatique de texte (*Natural Language Generation*, NLG) et la synthèse vocale (*Text To Speech*, TTS). Mis à part le cas des systèmes de dialogue génératifs que nous aborderons en Section 3.1.2.2 et en tenant compte du fait que la tendance actuelle soit à la fusion des modules de compréhension et de génération automatique de texte avec le gestionnaire de dialogue, tous les systèmes de dialogue actuels peuvent être vus selon ce paradigme.

Le dialogue humain/machine se déroule comme présenté sur la Figure 3.1. Au temps t, l'énoncé u_t de l'utilisateur est converti par l'ASR en une ou plusieurs séquences u_t^{ASR} de mots accompagnées d'un score de reconnaissance. Cette séquence est ensuite traitée par le SLU qui fournit un résumé du dialogue u_t^{SLU} (appelé contexte). À partir de ce contexte, le DM fournit une réponse appropriée o_t^{DM} sous la forme d'un acte de dialogue. Celui-ci est traduit en langage naturel par le NLG. Enfin, cet énoncé o_t^{NLG} est converti en signal sonore o_t à destination de l'utilisateur du système.

3.1.1.1 La reconnaissance vocale

La reconnaissance vocale (*Automatic Speech Recognition*, ASR) est le composant le plus bas-niveau d'un système de dialogue (avec son pendant à la sortie, la synthèse vocale). Son rôle est de convertir le signal audio en une liste des N hypothèses les plus

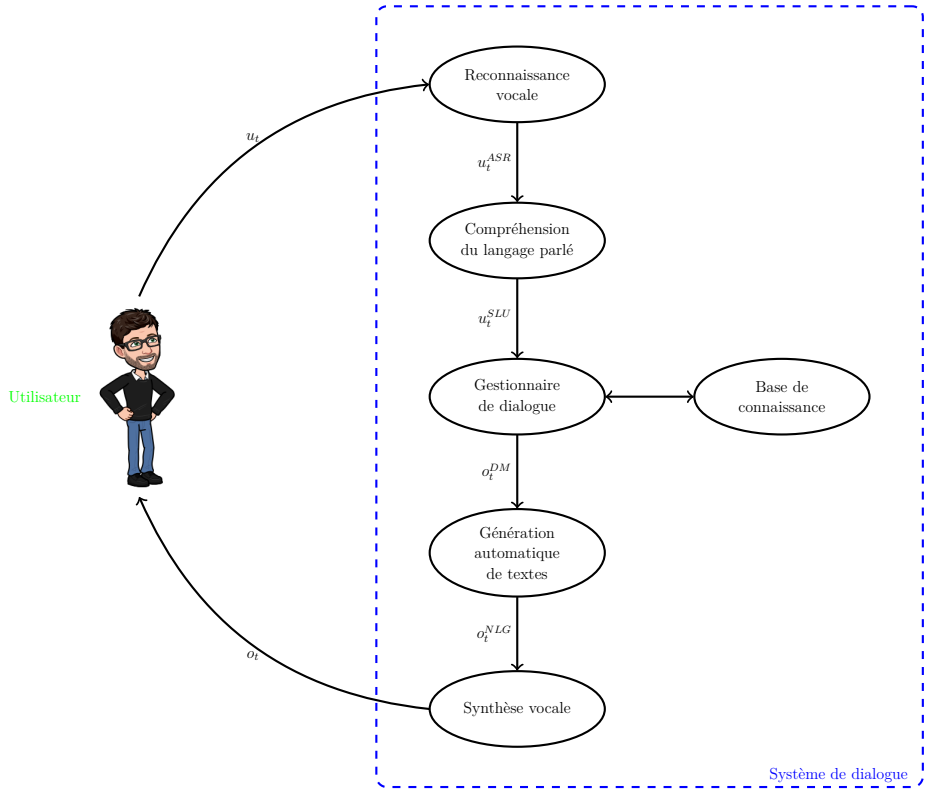


FIGURE 3.1 – Chaîne de dialogue

probables sur ce qui a été dit par l'utilisateur. Chacune de ces hypothèse peut se voir accompagnée d'un score de confiance (Rabiner and Schafer (2007)).

Par delà les systèmes de dialogue, les applications de l'ASR sont nombreuses. Il est intéressant de remarquer que l'ASR a été, dans les années 1970, la première application de l'apprentissage automatique aux interactions humain/machine (Jelinek et al. (1975)). L'ambition d'alors était de réaliser, non pas un système de dialogue, mais un système de dictée.

Il est remarquable de voir que le principe de base utilisé alors, le décodage de chaines de Markov cachées que nous présenterons au Chapitre 7 (Rabiner (1989)), est resté au coeur des systèmes de reconnaissance vocale jusqu'à très récemment (Gales and Young (2008)). On utilise cependant depuis Mohamed et al. (2009) des réseaux de neurones profonds (Deng et al. (2013)) fournissant de meilleures performances, capables même de rivaliser avec l'être humain (Amodei et al. (2016)).

3.1.1.2 La compréhension du langage parlé

Le premier rôle du module de compréhension est de fournir au gestionnaire de dialogue une représentation exploitable du contexte dialogique. La forme finale de ce contexte dialogique n'est pas rigoureusement définie car elle dépend de la manière dont la gestion du dialogue est conçue. Une contrainte d'interprétabilité est par ailleurs

souvent demandée. Dans ce cas, le rôle du module de SLU est d'extraire une sémantique à partir de la sortie de l'ASR (Tur and De Mori (2011)).

Dans ce cadre, une tâche très populaire dans la recherche en dialogue est le suivi de croyance (*Belief Tracking* en anglais). Ce problème se situe dans le cadre des dialogues orientés tâches. Initialement, le dialogue est représenté sous la forme d'un formulaire à trou. L'objectif du *Belief Tracking* est alors de compléter ce formulaire (remplir chaque trou et lui associer un score de confiance) au fur et à mesure de l'avancement de dialogue. Le *Dialogue State Tracking Challenge* est une compétition renommée de Belief Tracking (Williams et al. (2016)). Depuis leur introduction dans cette compétition dans Henderson et al. (2013), les réseaux de neurones récurrents se sont montrés être les modèles les plus appropriés à la résolution de la tâche (Henderson (2015)).

Le principal inconvénient des systèmes se basant sur un *Belief tracker* de ce type est la nécessité de construire par un expert humain un formulaire pour chaque tâche de dialogue, ce qui est susceptible de demander une certaine quantité de travail humain dans le cas d'un système multi-tâches. Par ailleurs, une limite de ce paradigme est qu'il n'est pas toujours possible de résumer un dialogue sous la forme d'un formulaire, comme dans le cas d'un dialogue de type *chit-chat*.

Cependant, dans le cas d'un gestionnaire de dialogue apprenant par renforcement, cette contrainte d'interprétabilité n'est a priori pas nécessaire. L'objectif est simplement d'obtenir une représentation markovienne et compacte. Ici encore, l'avènement des réseaux de neurones récurrents appliqués à l'apprentissage par renforcement (Hausknecht and Stone (2015)) permit de franchir une nouvelle étape vers la conception de systèmes automatisés requérant aussi peu d'expertise humaine que possible (Li et al. (2016), Barlier et al. (2018)). Des méthodes hybrides, telles que présentées dans Wen et al. (2017) combinent les deux approches précédentes en fournissant une sortie composée de la concaténation de la sortie du *belief tracker* et de l'état caché d'un réseau de neurones récurrent.

3.1.1.3 La gestion du dialogue

Le DM (pour *Dialogue Manager*) est, dans un système de dialogue, l'organe chargé de la gestion de l'interaction, de la manière de séquencer celle-ci. Il choisit quelle sera la future sortie du système. En d'autres termes, il choisit quoi dire à quel moment.

La sortie de ce module prend diverses formes. L'une des plus populaires à l'heure actuelle est de fournir un *acte de dialogue*. Ceux-ci, également appelés *actes de langage*, ont été introduits et théorisés dans Austin (1975) et Searle (1969). Ils représentent les actions d'un agent dialoguant lui permettant de changer l'état du monde. En effet, cette théorie est basée sur le fait que l'on dialogue non pas pour *dire* des choses, mais pour *faire* des choses : on dit alors que les énoncés sont *performatifs*, par opposition à des énoncés *descriptifs*. Un agent dialoguant ne cherche donc pas simplement à transmettre de l'information à son interlocuteur mais à changer son état interne, *i.e.* l'informer, le convaincre....

Cet acte de dialogue peut prendre diverses formes, il peut ainsi être par exemple

tiré d'un ensemble d'actes de dialogues préalablement sélectionné par un expert humain comme informer, annoncer, refuser, accepter... (Young et al. (2013)).Mais il peut également être un vecteur abstrait ayant été extrait de données (Wen et al. (2017)).

Une approche gagnant en popularité à l'heure actuelle consiste à fournir en sortie du DM directement les mots à énoncer (Li et al. (2016), De Vries et al. (2017)). Bien que ces approches soient extrêmement prometteuses, à l'heure actuelle, elles ont à notre connaissance uniquement été appliquées sur des tâches synthétiques. Dans ce cadre, ces modèles ont tendance à passer outre les modèles de langage, en profitant des failles du simulateur, pour résoudre la tâche de la manière la plus efficace possible, rendant délicate leur application à des utilisateurs réels.

La majeure partie de ce document étant consacrée au DM, nous y reviendrons dans la Section 3.2.

3.1.1.4 La génération automatique de textes

Le rôle de la génération automatique de textes (*Natural Language Generation*, NLG) est de transformer le signal de sortie du DM (généralement un acte de dialogue, éventuellement sous forme vectorielle) en une phrase en langage naturel.

Ainsi que remarqué dans Stent et al. (2005), un module de NLG doit être performant selon les critères suivants : adéquation avec l'acte de dialogue en entrée du module, maîtrise du langage cible et variabilité afin d'éviter les répétitions.

Traditionnellement (Stent et al. (2004)) l'acte de dialogue à l'entrée de la NLG est converti sous une forme intermédiaire de type arbre ou bien sous la forme d'un *template*. Celui-ci est ensuite traduit en langage naturel de manière à respecter les contraintes indiquées ci-dessus. Cependant, Wen et al. (2015a,b) ont montré que les modules de NLG basés sur des réseaux de neurones récurrents, particulièrement ceux du type LSTM, pouvaient se révéler particulièrement performants, tout en requérant moins d'expertise humaine que les modules traditionnels, permettant alors la conception de systèmes plus généraux. Dušek and Jurčíček (2016b,a) ont montré qu'il était possible de concevoir un système apprenant de manière purement statistique capable de fournir des réponses à la fois complexes, variées et adaptées au niveau de langage de l'interlocuteur.

3.1.1.5 La synthèse vocale

La synthèse vocale (*Text To Speech*, TTS) transforme la phrase générée par la NLG en un signal vocal qui sera audible pour l'utilisateur du système de dialogue (Rabiner and Schafer (2007)). Historiquement, la TTS est le premier composant d'un système de dialogue conçu par l'humain (Von Kempelen (1791)) et les premières méthodes statistiques pour la TTS sont arrivées très rapidement après celles pour l'ASR (Olive and Spickenagel (1976)).

Ces méthodes peuvent être de différentes formes. La synthèse vocale concaténative (*concatenative TTS* (Hunt and Black (1996)) par exemple, concatène différents fragments de voix humaine pour fournir le signal audio désiré. Les systèmes de synthèse vocale paramétriques quant à eux, traditionnellement basés sur les modèles de Markov

cachés, encodent les caractéristiques de la voix à générer sous la forme d'un certain nombre de paramètres (Zen et al. (2009)). Générer le signal vocal désiré se fait ensuite en jouant sur l'entrée du système.

L'architecture de réseau de neurones *WaveNet* (van den Oord et al. (2016, 2017)), basée sur un réseau de neurones de type génératif, a permis de dépasser les limites imposées par les deux précédents paradigmes, en fournissant un signal audio de qualité équivalente à la voix humaine tout en ayant la malléabilité des systèmes paramétriques.

3.1.2 Vers des systèmes *end-to-end*

3.1.2.1 Le besoin d'un système *end-to-end*

Bien que la majorité des systèmes de dialogue actuels puissent être représentés selon l'architecture modulaire présentée dans la Section 3.1.1, cette vision est simplificatrice. En effet, les différents modules ne sont pas indépendants et des boucles de rétroaction entre les modules sont généralement souhaitables.

Par exemple, la contextualisation produite par le module de compréhension peut servir à désambiguïser une phrase mal comprise par l'ASR, et de ce fait améliorer conséquemment ses performances.

Plus important encore, dans le cadre d'un DM apprenant par renforcement, il a été montré qu'apprendre conjointement représentation de l'espace d'états et politique permettait d'améliorer grandement les performances de l'apprentissage par renforcement (Strub et al. (2017), Wei et al. (2018)). Comme nous le verrons dans la Section 3.1.1.3, un DM apprenant par renforcement apprend à gérer le dialogue en fonction des récompenses lui ayant été préalablement fournies. Cependant, ces récompenses dépendent (au moins partiellement) de la satisfaction de l'utilisateur. Et l'utilisateur n'a jamais accès à la sortie du DM, seulement à la sortie du module de synthèse vocale. Ainsi, la modification du module de TTS ou de NLG devrait changer la politique du DM, ce qui ne peut pas être le cas dans l'architecture présentée. Mais une telle modification implique de réentrainer le DM à partir de zéro, ce qui n'est jamais souhaitable.

Les systèmes apprenant de bout en bout (appelés systèmes *end-to-end*) sont une approche prometteuse pour outrepasser ces limitations. Dans un tel système, l'espace d'état serait construit directement à partir du signal sonore et la sortie du DM serait le signal audio, généré échantillon par échantillon. De cette manière, les problèmes de reconnaissance vocale, de compréhension et de génération de paroles seraient résolus de manière implicite et les problèmes d'interdépendances cités précédemment sont automatiquement résolus.

La conception de tels systèmes est cependant un horizon à l'heure actuelle. L'une des raisons pour cela est que bien que les réseaux de neurones récurrents se soient montrés performants pour apprendre des dépendances long-termes, celles-ci seraient bien trop importantes dans le cadre d'un tel système pour pouvoir être traitées avec les modèles actuels. Par ailleurs, ce genre de système se comporterait *a priori* comme une boîte noire et la non-présence des modules ne permet pas de pouvoir améliorer

Interlocuteur 1	Salut
Interlocuteur 2	Salut. Comment vas-tu aujourd'hui ?
Interlocuteur 1	Je vais bien, merci. Comment vas-tu ?
Interlocuteur 2	Super, merci ! Je regarde Game of Thrones avec mes enfants.
Interlocuteur 1	Super ! Quel âge ont-ils ?
Interlocuteur 2	Ils sont quatre, allant de 10 à 21 ans. Et toi ?
Interlocuteur 1	Je n'ai pas encore d'enfants.
Interlocuteur 2	Tu peux donc garder tout le pop-corn pour toi.
Interlocuteur 1	Plutôt les Cheetos en ce moment.
Interlocuteur 2	Très bon choix !

FIGURE 3.2 – Exemple de dialogue de type *chit-chat*

spécifiquement l'un d'eux. Ainsi, augmenter le champ d'applications du système en augmentant son modèle de langage implique de tout devoir réentraîner.

3.1.2.2 Modèles génératifs

Une approche alternative basée sur des modèles statistiques génératifs gagne peu à peu du terrain sur l'architecture précédemment décrite. Celle-ci se présente comme une méthode *end-to-end* et propose d'aborder le problème du dialogue comme un problème de traduction. Dès lors, l'utilisation de ces systèmes serait pour obtenir des dialogues de type *chit-chat*. Contrairement aux dialogues orientés tâches ayant pour objectif la réalisation d'une tâche, les dialogues *chit-chat* n'ont pas d'objectifs précis et cherchent simplement à maximiser l'engagement de son utilisateur, *i.e.* le faire interagir le plus possible et le plus longtemps possible avec son système (Ritter et al. (2011)). La Figure 3.2 présente un exemple de dialogue de type *chit-chat* extrait du corpus PERSONA-CHAT et traduit en français (Zhang et al. (2018)).

Cette approche a été proposée pour la première fois dans Ritter et al. (2011) pour aborder la génération de réponses (en anglais *Response Generation*, RG). Ce problème ci apparaît comme faisant le lien entre le problème de réponse aux questions (en anglais, *Question Answering*, QA) et celui de dialogue. Il peut être défini de la manière suivante : ayant un certain stimulus conversationnel, répondre de manière approprié. Ainsi, la génération de réponses est similaire à la réponse aux questions car seul le futur à court terme est considéré, il s'agit ici de répondre la bonne chose au bon moment. Cependant, la forme de la réponse n'est pas aussi guidée que dans le cas du *Question Answering*. Leur approche montre qu'il était possible d'obtenir des réponses cohérentes en *traduisant* un poste Twitter en une réponse plausible.

Ce travail est étendu dans Sordoni et al. (2015), où le contexte dialogique est introduit dans le modèle et est implicitement appris par le biais d'un réseau de neurones récurrent. La réponse générée est donc maintenant conditionnée par le passé. Ainsi, même si la problématique de planification n'est toujours pas abordée, le problème se rapproche de plus en plus du problème de dialogue. Cette approche est également

choisie par Vinyals and Le (2015) qui utilisent une architecture de type *Seq2Seq* pour *traduire* une phrase en sa réponse.

En raison des promesses de concevoir un modèle de conversation apprenant de bout en bout à partir de données brutes et non pas à partir de règles, ces méthodes ont été très populaires au moment de leur sortie. Elles se heurtent cependant aux limitations suivantes : les objectifs du dialogue humain/humain ne sont pas introduits dans le modèle et il est impossible d'assurer une cohérence grammaticale.

Selon ce paradigme, on ne cherche pas à optimiser une certaine fonction de récompense mais plutôt à apprendre les probabilités de transition d'une certaine chaîne de de Markov. Plus exactement, on ne va pas apprendre cette distribution mais chercher à échantillonner selon cette distribution. Autrement dit, on va chercher à dire la bonne chose au bon moment sans se préoccuper de la suite de la conversation. De telles approches sont dites *génératives* et rentrent dans le cadre de l'apprentissage par imitation (Daumé et al. (2009)).

Cette méthode est très efficace pour générer des données cependant, les applications d'un DM conçu de cette manière se trouveront en nombre très limité. Tout d'abord, ne pas se préoccuper de la suite de la conversation peut rendre très difficile la résolution de certains problèmes de dialogues orientés tâches. Ensuite, pouvoir apprendre des discussions ne peut se faire que si l'on dispose de conversations expertes or jusqu'à présent, ces conversations ne correspondent qu'à des dialogues humain/humain qui, comme nous l'avons vu, sont différents. Il n'y a par exemple pas de gestion des erreurs de l'ASR et du SLU. Ces méthodes sont par ailleurs également très gourmandes en données. Enfin, les erreurs générées par un tel système ont tendance à s'accumuler, ce qui aura tendance à orienter la conversation vers des états non présents dans les données et dans lesquels le système de dialogue ne sera pas performant, voir ne sera pas capable de se comporter de manière cohérente (Ross et al. (2011)).

3.2 Apprentissage par renforcement pour la gestion du dialogue

La tâche de gestion du dialogue peut être vue comme trouver quoi dire connaissant le contexte dialogique, tout en sachant que ce qui sera dit impactera le futur de la conversation. Cette tâche peut ainsi être vue comme un problème de décision dans l'incertain, un incertain dû à l'impossibilité de définir exactement le contexte dialogique et à la variabilité du comportement des utilisateurs.

Comme explicité dans la section précédente, l'apprentissage par renforcement est la réponse de l'apprentissage automatique à ce problème de décision. Cette observation fut faite pour la première fois par Levin and Pieraccini (1997), l'article fondateur formalisant la gestion du dialogue humain-machine et plus généralement des interactions humain-machine comme un problème d'apprentissage par renforcement. Les bénéfices d'un tel paradigme sont nombreux : il n'y a plus besoin de concevoir des stratégies de dialogue complexes et très dépendantes du domaine d'application, le système apprend

continuellement ce qui permet d'éviter de repartir de zéro à l'ajout d'une nouvelle fonctionnalité et la gestion des erreurs par le système gagne en précision. Depuis, l'utilisation de l'apprentissage par renforcement est devenu une norme pour la gestion du dialogue.

Se représenter un problème sous la forme d'un MDP implique de définir un espace d'états, un espace d'actions et une fonction de récompense, la fonction de transition étant définie implicitement par l'espace d'états. Dans les prochaines sections, nous allons chercher à les définir de la manière la plus précise possible et nous discuterons des implications de chaque choix. Enfin, nous discuterons des avantages et inconvénients des différents algorithmes proposés dans la littérature pour résoudre ce problème.

3.2.1 L'espace d'états

Comme explicité en Section 3.1.1.2. La question de la représentation d'états est très proche de la question de quel signal doit se trouver en sortie du SLU. Dans le cadre d'un DM apprenant par renforcement, il s'agit de trouver une représentation d'état à la fois markovienne et efficace. Le *TRINDI Information State* a été l'une des premières tentatives de normalisation de représentation de l'état du dialogue (Larsson and Traum (2000)). Cette représentation n'a pas été proposée pour l'apprentissage par renforcement, mais l'idée principale est là : on agit selon un état (peu importe la manière dont nous avons sélectionné notre action), notre environnement réagit alors ce qui modifie notre état, puis nous recommençons jusqu'à ce que le dialogue se termine.

3.2.1.1 Processus de Markov partiellement observables

Pour obtenir une représentation d'état markovienne du dialogue, connaître l'état interne de l'utilisateur, *i.e.* ses connaissances et son objectif, ainsi que l'historique de la conversation est suffisant. Cette représentation de l'avancée d'une conversation est par ailleurs facilement interprétable. Cependant, ces données ne sont pas accessibles directement. En effet, l'état interne de l'utilisateur ne peut être que supposé, ses énoncés arrivent par ailleurs bruités au Dialogue Manager.

Partant de ce constat, Roy et al. (2000), Williams and Young (2007) choisissent de représenter l'état du dialogue comme une distribution de probabilité sur les états internes de l'utilisateur et sur les historiques de conversation possibles, se plaçant ainsi selon le paradigme des processus décisionnels de Markov partiellement observables (ou en anglais *Partially Observable Markov Decision Process*, POMDP). Cette approche a l'avantage d'améliorer grandement la gestion des erreurs de la reconnaissance vocale et du module de compréhension. Cependant, son point faible est sa grande complexité computationnelle.

La solution proposée dans Young et al. (2010) pour pallier les problèmes de *tractabilité* de l'approche précédente, le *Hidden Information State*, est de construire une liste des N situations les plus probables connaissant les observations. Construire cette liste place donc le problème de la caractérisation de l'état courant comme un problème de *Belief Tracking*, exposé en Section 3.1.1.2.

3.2.1.2 Apprentissage de l'espace d'états

Bien que l'approche exposée au paragraphe précédent a connu de nombreux succès pour la conception de systèmes de dialogue, elle n'est pas dénuée de défaut. Le principal inconvénient de cette approche est que la liste d'hypothèses correspondant à l'état du dialogue a été préalablement conçue par un être humain. Cette représentation peut donc manquer d'expressivité et ne pas respecter la propriété de Markov en raison des choix et des erreurs commises par ce concepteur.

Les représentations d'états apprises, ne se basant pas sur l'expertise humaine, présentent l'avantage de résoudre ce problème. En raison de leur manque d'interprétabilité, elles ne furent adoptées que récemment. Daubigney et al. (2013) proposent d'apprendre une représentation d'état par l'intermédiaire des *Echo State Networks* (ESN) tandis que El Asri et al. (2017) suggèrent de résoudre ce problème par l'intermédiaire d'algorithmes génétiques appliqués sur une mémoire distribuée à basse densité.

Les récentes avancées dans le domaine du *Deep Reinforcement Learning* ont permis l'implémentation d'algorithmes apprenant cette représentation d'état de manière efficace. Ainsi, Zhao and Eskenazi (2016) apprennent l'espace d'état en même temps que la politique avec DRQN tandis que Li et al. (2016), Barlier et al. (2018) proposent d'utiliser l'état interne d'un réseau de neurones récurrent ayant appris à générer la réponse la plus probable à l'énoncé de l'utilisateur.

Un deuxième intérêt d'une représentation d'états apprise permet la gestion efficace d'un système de dialogue multi-modal dans lequel les entrées prennent la forme d'un énoncé mais peuvent être accompagnées d'une image par exemple (Strub et al. (2017)).

3.2.2 L'espace d'actions

Comme exposé au paragraphe 3.1.1.3, l'espace d'actions du DM peut prendre différentes formes. Dans le cas le plus courant, l'espace d'actions correspond à un ensemble fini de N actes de dialogues. Un compromis se pose alors dans la question de quels actes proposer au système de dialogue. En proposer trop reviendrait à rendre le problème trop demandant en temps de calcul mais en proposer trop peu reviendrait à contraindre énormément les capacités du système de dialogue. Diverses stratégies existent pour restreindre à chaque tour l'espace d'actions de la machine. Williams (2008) propose de d'implémenter un ensemble de règles présélectionnant un ensemble d'actions, tandis que Serban et al. (2017), Peng et al. (2017) proposent une approche hiérarchique où un contrôleur haut-niveau sélectionne une action parmi un ensemble d'actions proposées par différents contrôleurs bas-niveau.

Il se peut aussi que la sortie du DM soit un vecteur abstrait correspondant à une *intention* (Wen et al. (2017)). Dans ce cas, l'espace d'action du système est continu et une difficulté se pose alors dans la manière traduire cette intention en une phrase intelligible. He et al. (2015) proposent dans ce cas de choisir parmi un ensemble de N phrases celle étant la plus proche de cette intention.

Dans un troisième cas, l'espace d'actions du DM est l'ensemble des mots d'un certain vocabulaire (Li et al. (2016), De Vries et al. (2017)). Dans ce cadre, le DM effectue

plusieurs actions avant de relâcher la parole.

Enfin, il est important de noter que dans certains cas comme les systèmes industriels, l'action du système de dialogue peut ne pas être à destination de l'utilisateur. De telles actions sont par exemple des actes d'écritures ou de lecture dans une base de données ou une base de connaissances (Eric et al. (2017), Dhingra et al. (2017)). Ces actions peuvent être nécessaires au succès du dialogue mais ne relèvent pas immédiatement du dialogue en temps que tel.

3.2.3 La fonction de récompense

Dans le cas d'un gestionnaire de dialogue apprenant par renforcement, la question de la définition de la fonction de récompense est étroitement liée avec la question de l'évaluation de la performance d'un système. En effet, une fois qu'un critère de performance est défini, il est tout à fait possible d'utiliser ce critère comme la fonction de récompense à optimiser.

Il n'est cependant pas évident de définir la performance d'un système de dialogue. Du point de vue de l'utilisateur, sa satisfaction à la fin du dialogue peut généralement être vue comme l'ultime métrique à optimiser. Du point de vue du concepteur du système cependant, celle ci n'est facilement définissable et automatisable.

Si l'on considère un unique dialogue orienté tâche, il est raisonnable de penser que la performance peut être définie comme sa capacité à résoudre efficacement cette tâche. Cependant, si la satisfaction utilisateur est corrélée avec ce critère de performance, il n'est pas possible de complètement les identifier. Walker et al. (1997) proposent de définir la performance d'un système de dialogue en régressant linéairement la satisfaction utilisateur (mesurée par un expert ou bien en faisant remplir un questionnaire à la fin de chaque dialogue) par rapport à différentes caractéristiques mesurées indépendamment : l'efficacité du dialogue (mesurée en nombre de tours moyen par dialogue), la qualité du dialogue (la qualité des phrases prononcées par le système de dialogue), et la résolution de la tâche. De façon intéressante, Walker et al. (2001) ont montré que le coefficient affecté à la longueur du dialogue était négatif, montrant que de manière générale, les utilisateurs préfèrent un dialogue long à un dialogue extrêmement court. Bien que cette approche ait connu de grands succès et de nombreux développement (Möller and Ward (2008)), son principal point faible est qu'elle est applicable uniquement dans le cadre d'un dialogue orienté tâche. Si la tâche change, il se peut que les coefficients affectés à chaque caractéristique puisse également changer. Par ailleurs, le caractère exhaustif de la liste des caractéristiques prises en compte ne pourra jamais être démontré.

L'apprentissage par renforcement inverse (*Inverse Reinforcement Learning*, IRL) apporte une solution à ce problème (Ng et al. (2000), Neu and Szepesvári (2009)). Dans ce cadre, le problème posé est le suivant : si j'observe un certain comportement (supposé expert), quelle est la fonction de récompense qu'il optimiserait si il était un MDP ? Piot et al. (2014b) montre l'intérêt d'une telle approche dans un cadre non loin de celui du dialogue. Son objectif est d'apprendre à une machine à rire de la même

manière qu'un être humain. L'approche par renforcement classique semble très difficile. Évaluer les performances de la machine dans une telle situation est un exercice délicat. L'apprentissage par renforcement inverse, lui, permettra à la machine de rire là où les êtres humains auraient ri. De la même manière, apprendre la fonction de récompense d'un DM par le biais de l'apprentissage par renforcement inverse permet de définir un comportement idéal tout en évitant de se poser la question de ce qu'est un dialogue réussi (Pietquin (2013), Chinaei and Chaib-draa (2014)).

El Asri et al. (2014, 2016b) adoptent une approche différente de l'apprentissage de fonction de récompense. Ici, l'idée est non pas de chercher à trouver la récompense imitant au mieux l'expert mais de trouver la fonction de récompense correspondant au mieux aux évaluations des utilisateurs humains.

3.2.4 Apprentissage de stratégies de dialogue

L'implémentation d'un système de dialogue basé sur l'apprentissage nécessite un grand nombre de données, et ce quel que soit le module considéré. Le comportement humain est en effet à la fois très variable selon les individus et complexe. Les approches combinatoires, visant à intégrer le dialogue dans un arbre de décision sont donc limitées à des dialogues très simples (du type réserver un restaurant ou prendre rendez-vous chez le coiffeur). Idéalement, un gestionnaire de dialogue pourrait outrepasser cette limitation en apprenant par renforcement via une expérience directe avec des utilisateurs. Malheureusement, celle-ci est trop coûteuse pour que ce puisse être le cas.

Un système de dialogue performant doit être capable de gérer un nombre limité de données. Malheureusement, comme nous le verrons dans la Section 3.2.4.2, un algorithme suffisamment efficace pour gérer cette parcimonie n'existe pas encore. Pour pallier ce problème, une solution commune est d'utiliser le maximum de données présentes pour réaliser un simulateur d'humain, de faire apprendre au DM une stratégie élémentaire à partir de ce simulateur, puis de raffiner cette stratégie en interagissant avec de véritables êtres humains.

3.2.4.1 Simulation d'utilisateurs

Un simulateur d'utilisateur est un ersatz d'humain ayant supposément un comportement similaire à celui-ci (Pietquin and Hastie (2013)). Comme exposé à la section précédente, l'entrée du gestionnaire de dialogue n'est généralement pas le signal audio. Il est donc courant de concevoir un simulateur tel que sa sortie soit sous la forme de la sortie d'un SLU. Un simulateur d'utilisateurs est donc généralement composé d'un générateur de phrases (reflétant une phrase ayant pu être dite par un humain) et d'un simulateur de reconnaissance vocale (Pietquin and Renals (2002,?), Pietquin and Dutoit (2006b)) et de SLU (Pietquin and Dutoit (2006a)). De cette manière, un bon simulateur d'utilisateurs reflète non seulement le comportement humain, mais également les imperfections des modules de reconnaissance vocale et de compréhension.

Interagir avec un simulateur est moins coûteux et moins risqué qu'avec un utilisateur réel. Une fois que le système aura appris une politique de dialogue décente à l'aide de la

simulation d'utilisateurs, il est possible de vérifier la cohérence de son comportement par des experts puis d'affiner son comportement en interagissant avec de véritables humains.

Définir ce qu'est un simulateur d'utilisateur performant n'est cependant pas aisé. Idéalement, le rôle d'un simulateur est de se comporter d'une manière similaire à un être humain. Mais comment évaluer cette similarité ? Pietquin and Hastie (2013) listent les caractéristiques d'une métrique adaptée à l'évaluation d'un simulateur d'utilisateurs. Celle-ci doit être cohérente, tant d'un point de vue local (une réponse doit être adaptée à la question) que global (l'enchaînement des actes de dialogue doit être cohérent), refléter l'objectif de l'utilisateur, prédire les performances du système face à des utilisateurs réels, généraliser à des situations non présentes dans les données, être indépendante de la tâche et du système de dialogue à entraîner et finalement pouvoir être calculée de manière automatique. Chacun de ces critères peut être facilement évalué à l'aide de métriques connues tels la divergence KL, le score BLEU, la précision et le rappel... Cependant, la question de savoir si il existe une métrique combinant l'ensemble des critères exposés précédemment est toujours ouverte.

Le premier simulateur d'utilisateurs implémenté à partir de méthodes statistiques fut proposé dans Eckert et al. (1997). Ce simulateur générait des actes de dialogue à partir d'une distribution de probabilité conditionnée par la dernière action du système. Par la suite, les simulateurs ont été affinés en ne conditionnant plus uniquement par rapport à la dernière action du système mais également par rapport à l'objectif de l'utilisateur (Scheffler and Young (2001)) ou alors en considérant l'intégralité de l'historique de la conversation, encodé par exemple sous la forme de réseaux bayésiens (Pietquin and Dutoit (2006a)), d'une chaîne de Markov cachée (Cuayáhuitl et al. (2005)) ou bien basé sur un agenda (Schatzmann et al. (2007), Schatzmann and Young (2009)). Plus récemment, El Asri et al. (2016a) proposent une méthode basée sur les architectures de réseaux de neurones de type *Seq2Seq* surpassant les approches précédentes dans le réalisme du comportement de l'utilisateur simulé.

Parmi les approches précédentes, celles fonctionnant par apprentissage se situent dans le cadre de l'apprentissage par imitation. Dans ce cadre, on cherche à retrouver quelle réponse de l'utilisateur est la plus vraisemblable en fonction du comportement du système de dialogue. Cette probabilité est cependant apprise à partir de données. Dans le cas où le système de dialogue a un comportement non conventionnel, il est fort probable que l'on ne puisse pas retrouver de comportement humain plausible directement dans le jeu de données. Chandramohan et al. (2011) proposent d'aborder le problème de la génération d'utilisateurs simulés comme un problème d'apprentissage par renforcement. Dans ce cadre, l'utilisateur agit de manière à remplir ses objectifs. Ces objectifs ne sont a priori pas définis explicitement. Ils doivent donc être appris. La méthode proposée est alors d'apprendre grâce à l'apprentissage par renforcement inverse les préférences de l'utilisateur puis de résoudre le problème de décision défini par cette nouvelle fonction de récompense.

3.2.4.2 Apprentissage en ligne

Un système apprenant en ligne optimise sa politique par le biais d'interactions avec des utilisateurs humains ou un simulateur. Comme abordé en Section 2.2.5.1, un compromis doit constamment être réalisé entre agir d'une manière connue pour être efficace (phase d'*exploitation*) ou alors chercher à trouver de nouveaux meilleurs comportements (phase d'*exploration*).

Ce problème de gestion de l'exploration est difficile. En effet, on ne peut, par définition, savoir ce qui se passera après avoir agit d'une manière nouvelle. Mais il est par ailleurs nécessaire de pouvoir garantir de la part du système de dialogue des comportements sûrs, il n'est par exemple pas acceptable qu'un système de dialogue puisse avoir des paroles offensantes. Combiner ces deux besoins reste un véritable défi pour la recherche à l'heure actuelle. Nous aborderons dans le Chapitre 5 de quelle manière celui-ci peut être abordé.

Comme explicité en Section 2.2.5, deux types d'approches peuvent être envisagés pour faire de l'apprentissage par renforcement en ligne. Dans un premier cas, l'apprentissage est fait de manière *on-policy*. Autrement dit, on cherche à évaluer directement la politique suivie afin de l'améliorer. Dans ce cadre, des systèmes de dialogue basés sur les processus gaussiens (Gašić et al. (2010)) ou bien sur des architectures de type *natural actor-critic* (Jurčíček et al. (2010)) ont été implémentées avec succès.

Cette approche présente cependant l'inconvénient d'impliquer un grand nombre de mise à jour de la politique suivie. Contrôler alors la sûreté du système de dialogue est difficile. Les approches de type *off-policy*, elles, permettent d'évaluer les Q-valeurs de la politique optimale de manière indépendante de la politique suivie. Ainsi, une politique sûre (possiblement conçue manuellement) peut être utilisée pour collecter des données tout en apprenant en ligne à optimiser cette politique. Dans ce cadre, les différences temporelles de Kalmann (Pietquin et al. (2011b)) ont permis de réaliser des systèmes de dialogue performants. Le problème de savoir quand passer à la politique apprise se pose cependant.

3.2.4.3 Apprentissage hors-ligne

Apprendre la gestion de dialogue hors-ligne présente un certain nombre d'avantages par rapport à l'apprentissage en ligne décrit au paragraphe précédent. Tout d'abord, apprendre directement à partir d'un jeu de données récolté préalablement plutôt qu'apprendre par interaction permet d'éviter les problèmes d'exploration désastreuse cités précédemment. En effet, dans ce cadre, il est parfaitement possible de récolter des données à partir d'une politique sûre, puis d'optimiser cette politique à partir de celles-ci.

Par ailleurs, l'interaction avec les humains est coûteuse. Récolter un jeu de données, puis apprendre à partir de ce jeu permet une meilleure "rentabilité" de celle-ci. Aussi, il est plus aisé de comparer différents systèmes ayant appris sur la même base pour ne pas que les différences de performances de ceux-ci soient dues aux variations de comportement des utilisateurs.

Enfin, mettant à jour de manière synchrone les Q-valeurs et tirant parti des grandes

capacités de généralisation des algorithmes d'apprentissage supervisé, les algorithmes d'apprentissage par renforcement hors-ligne sont plus stables et plus rentables en termes de données que leurs alternatives en ligne.

Les deux grandes familles d'algorithmes d'apprentissage par renforcement hors-ligne, basés sur les algorithmes *LSPI* et *Fitted-Q*, furent respectivement implémentées avec succès dans le cadre de l'apprentissage de la politique pour la gestion du dialogue dans Li et al. (2009) et Pietquin et al. (2011c).

3.2.4.4 Approches hybrides

Les récents systèmes de dialogue basés sur le *Deep Reinforcement Learning* adoptent en majorité des approches hybrides, à mi-chemin entre l'apprentissage en ligne et l'apprentissage hors-ligne. Trois approches sont généralement à la base de ces sytèmes. Certains comme Cuayáhuitl et al. (2015), Li et al. (2017) sont basés sur DQN (Mnih et al. (2015)), d'autres comme Wen et al. (2017), Strub et al. (2017), Li et al. (2016) sont basés sur des méthode de gradient sur les politiques (en anglais *policy gradient*) de type REINFORCE (Williams (1992)). Enfin, les derniers Su et al. (2017) se basent sur un algorithme à mi-chemin entre les deux précédents, les architectures acteur-critique (Konda and Tsitsiklis (2003)).

Ces algorithmes ont tous besoin de l'interaction en ligne avec des utilisateurs pour fonctionner. Cependant, du fait de leur grand besoin en données et du besoin de stabilité, une réutilisation des échantillons collectés est toujours nécessaire. À l'heure actuelle, du fait de leur grande demande en données, ces méthodes ont uniquement été testées sur simulateur.

Deuxième partie

Contributions

Chapitre 4

Modéliser le dialogue humain/machine comme un jeu stochastique

Au Chapitre 3, nous avons vu que dans le cas d'un système de dialogue parlé, le processus de décision choisissant les énoncés du système est implémenté dans le Dialogue Manager. Par ailleurs, nous avons vu que dans le cas des dialogues orientés tâches, l'apprentissage par renforcement permet de mener à bien l'optimisation du comportement du DM, en dépit de la très forte stochasticité des comportements utilisateurs.

Rappelons brièvement que dans ce cadre, le dialogue est modélisé sous la forme d'un processus de décision de Markov (éventuellement partiellement observable) (Levin and Pieraccini (1997), Lemon and Pietquin (2007), Young et al. (2013)). À partir d'un contexte dialogique imparfait (dû aux erreurs de transcription des modules d'ASR et de SLU), le DM choisit, en fonction de son expérience, l'acte de dialogue qu'il estime être le plus approprié. L'être humain est alors vu comme une distribution de probabilité stationnaire, échantillonnant ses énoncés en fonction de l'historique du dialogue. Cette représentation du dialogue est cependant limitée, nous allons voir dans ce chapitre en quoi.

4.1 Motivations

Les gestionnaires de dialogue basés sur l'apprentissage par renforcement orientés ne considèrent en général que des dialogues orientés tâches. De plus, ils font l'hypothèse que l'utilisateur adopte un comportement non stationnaire. Cette hypothèse impose, entre autres, que l'utilisateur se s'adapte pas au système avec lequel il interagit. Modéliser un problème sous la forme d'un MDP ne peut donc que s'appliquer aux problèmes de type agent/environnement, aussi connus sous le nom de jeux contre la nature (Milnor (1951)). Dans ces problèmes, l'agent évolue dans un environnement non-stationnaire (ne changeant pas au cours du temps) et agissant de manière totalement désintéressée. La modélisation du dialogue sous la forme d'un MDP n'est donc valable que si :

— l'utilisateur ne modifie pas son comportement au cours du temps (la stratégie du DM est donc apprise dans un environnement stationnaire)

— le dialogue est orienté tâche et nécessite une collaboration entre l'utilisateur et la machine pour atteindre un objectif (qui est généralement celui de l'utilisateur).

Cette première hypothèse n'est pas respectée si l'utilisateur adapte son comportement à l'amélioration continue de son système de dialogue. Bien que des travaux de

modélisations sur la co-adaptation humain/machine existent (Bourguin et al. (2001)), ceux-ci ne peuvent pas s'étendre directement à l'apprentissage automatique et ne sont pas applicables à la conception systèmes de dialogue. Dans le cadre de l'apprentissage machine, une première tentative de modélisation des effets de cette co-adaptation a été proposée dans Chandramohan et al. (2012b). Cependant, dans ces travaux, l'humain et la machine sont considérés comme des agents indépendants. Aucune garantie de convergence ne peut être obtenue par cette approche.

La deuxième hypothèse est également contraignante. Un système de dialogue dit complet devrait être capable de mener à bien différents types de dialogue tels la résolution d'une tâche mais également la négociation (El Asri et al. (2014)), l'e-learning... Dans ces derniers cas, cette hypothèse d'objectif commun est violée. Ces dialogues ne peuvent donc se modéliser comme des MDP. Des travaux récents ont cependant adapté le paradigme des MDPs pour traiter ces dialogues. Ainsi, dans Georgila et al. (2014), le problème de la négociation est modélisé comme un problème d'apprentissage par renforcement multi-agent (MARL). Cependant, cette approche repose sur des algorithmes qui traitent la question multi-joueurs en tant que problème non-stationnaire (par exemple WoLF-PHC (Bowling and Veloso (2002))). Dans ce cadre, chaque agent est supposé maintenir une politique d'interaction stable pendant un temps suffisamment long pour que l'autre agent puisse en apprendre davantage sur sa politique actuelle. Ainsi, il n'y a jamais de garantie de convergence. Par ailleurs, dans le cas réel, il n'existe pas de raison pour laquelle l'être humain attende que la machine se soit adaptée à lui pour évoluer. Dans Efstathiou and Lemon (2014) et Lewis et al. (2017), les auteurs suivent la même direction en considérant que chaque agent agit dans un MDP stationnaire en ne se préoccupant pas des adaptations de l'autre agent.

Afin de pouvoir traiter des dialogues ne respectant pas ces hypothèses, nous proposons dans ce chapitre un changement de paradigme. Au lieu de modéliser le problème du dialogue comme un MDP, nous proposons de le voir comme un problème de jeu stochastique (Shapley (1953)). Ce paradigme étend celui des MDPs aux interactions multi-joueurs et permet d'apprendre ensemble les stratégies des deux agents (ici l'utilisateur et le DM). Ainsi, la stratégie apprise est la meilleure face à l'utilisateur (un MDP apprendrait la meilleure stratégie par rapport à un utilisateur type défini comme une *moyenne* des utilisateurs rencontrés). Ce paradigme présente l'avantage de modéliser à la fois la co-adaptation et la non-coopération possible. Contrairement aux modèles basés sur la théorie des jeux standard (Caelen and Xuereb (2011)), les jeux stochastiques permettent d'apprendre à partir de données. Plus particulièrement, en partant des résultats de Perolat et al. (2015), nous montrons que la stratégie optimale peut être apprise à partir d'un ensemble de données collectées de manière hors-ligne comme pour les MDP (Pietquin et al. (2011a)). Ainsi, des politiques de négociation optimales peuvent être apprises à partir d'interactions non optimales. Ce nouveau paradigme est également très différent des méthodes MARL proposées dans les travaux antérieurs (Chandramohan et al. (2012b), Georgila et al. (2014), Efstathiou and Lemon (2014)) puisque l'optimisation est réalisée conjointement et alternativement pour chaque agent.

4.2 Les jeux stochastiques

Les jeux stochastiques (Filar and Vrieze (1996)), introduits pour la première fois dans Shapley (1953), sont l'extension naturelle des MDPs au cadre multi-agents.

4.2.1 Définitions

Formellement, un jeu stochastique se définit de la façon suivante :

Définition 4.1. *Un jeu stochastique (en anglais Stochastic Game, SG) est un 6-uplet* $\langle \mathcal{D}, \mathcal{S}, \boldsymbol{A}, T, \boldsymbol{R}, \gamma \rangle$ *avec :*

- $\mathcal{D} = \{1, ..., n\}$ *représente l'ensemble des agents*
- $\mathcal{S}$ *est l'espace d'états*
- $\boldsymbol{A} = \times_{i \in \mathcal{D}} \mathcal{A}_i$ *est l'ensemble des actions joints, où pour tout* $i = 1, ..., n$, $\mathcal{A}_i$ *est l'espace d'actions de l'agent* i
- $\boldsymbol{R} = \times_{i \in \mathcal{D}} \mathcal{R}_i$ *est la fonction de récompense jointe, où pour tout* $i = 1, ..., n$, $\mathcal{R}_i : \mathcal{S} \times \boldsymbol{A} \to \mathbb{R}$ *est la fonction de récompense de l'agent* i
- $T : \mathcal{S} \times \boldsymbol{A} \times \mathcal{S} \to [0, 1]$ *est la fonction de transition*
- $\gamma \in [0, 1)$ *est un facteur d'amortissement*

Comme présenté en Figure 4.1, l'interaction entre les agents et l'environnement dans le cadre des jeux stochastiques est très similaire au cadre des MDPs. Au temps t, chaque agent i est dans un état s_t (commun à l'ensemble des agents). Il exécute alors une action a_t^i. On notera $\mathbf{a}_t$ l'action jointe de l'ensemble des agents. Réaliser cette action fait réagir l'environnement qui, au pas de temps suivant, met les agents dans l'état s_{t+1} et fournit à chacun une récompense r_t^i.

Un agent i choisit dans l'état s une action a_i en fonction de sa *stratégie* σ_i. Dans le cas général, cette stratégie est une distribution de probabilité sur l'espace d'actions associé à l'état s. Dans le cas où l'on considère l'ensemble des agents, nous parlons de *stratégie jointe*. La notation σ_{-i} fait référence à la la stratégie jointe ôtée de la stratégie de l'agent i.

Ainsi, chaque "MDP" dans lequel plusieurs agents interagissent peut être vu comme un jeu stochastique. Un jeu stochastique avec uniquement deux agents dotés de fonctions de récompense opposées (*i.e.* $\mathcal{R}_1 = -\mathcal{R}_2$) est dit *à somme nulle* (ou *purement compétitif*). À l'opposé, un jeu dans lequel les fonctions de récompense sont identiques est dit *purement coopératif*. Enfin, un jeu n'étant ni à somme nulle, ni purement coopératif est dit à somme générale.

4.2.2 Meilleure réponse

Dans le cadre agent/environnement, l'agent apprend en adaptant son comportement aux situations précédemment rencontrées. Cela est également valable dans un scénario multi-agent, si l'agent i veut en savoir plus sur l'agent j, il agira selon ce qui a déjà été appris à propos de j. Mais à l'inverse, si j veut en savoir plus sur l'agent i, il agira

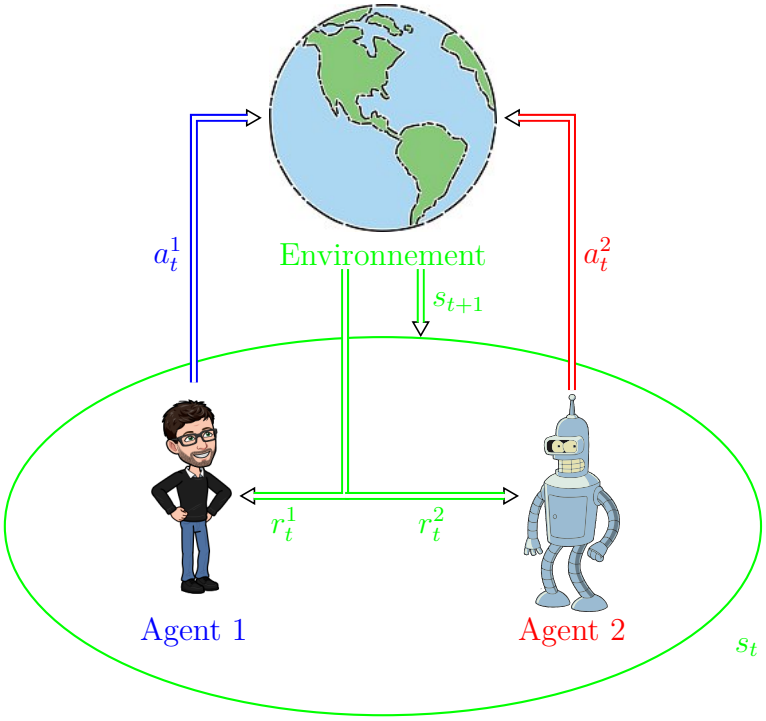


FIGURE 4.1 – Jeu Stochastique

selon ce qu'il sait de i. Nous disons que les agents co-adaptent. La co-adaptation est, en raison de cette boucle de rétroaction, un processus intrinsèquement non stationnaire. On dit d'un algorithme qu'il converge si la politique de chaque agent converge vers une stratégie stationnaire.

Comme dans le cadre mono-agent, chaque agent agit de manière à maximiser la somme espérée de ses récompenses futures, aussi appelée *valeur de la stratégie jointe* $\boldsymbol{\sigma}$, définie dans l'état s pour l'agent i par : $V_i(s, \boldsymbol{\sigma}) = E[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{\sigma})]$. Une Q-fonction peut alors également être définie par :

$$Q(s, \boldsymbol{\sigma}, \mathbf{a}) = R(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \mathbf{a}, s') V(s', \boldsymbol{\sigma})$$

Cette fonction de valeur dépend des stratégies de tous les agents environnants. Dans le cas général, il n'est donc pas possible de définir une stratégie optimale dans tous les cas. Une définition plus faible de l'optimalité se doit donc d'être considérée. Une meilleure réponse (en anglais *Best Response*, BR) est une stratégie optimale sachant celles des autres agents. Formellement,

Définition 4.2. *On dit que l'agent i applique une meilleure réponse σ_i contre la stratégie jointe des autres agents σ_{-i} si σ_i est optimale sachant σ_{-i}. On écrit alors $\sigma_i \in BR(\sigma_{-i})$.*

La définition d'équilibre de Nash, remplaçant la notion d'optimalité des MDPs, vient alors naturellement :

Définition 4.3. *L'ensemble de stratégies $\{\sigma_i\}_{i\in\mathcal{D}}$ est un équilibre de Nash (en anglais Nash Equilibrium, NE) si pour chaque $i \in \mathcal{D}$, nous avons $\sigma_i \in BR(\sigma_{-i})$.*

Ainsi, on dit qu'une stratégie jointe est un équilibre de Nash si chaque stratégie est optimale sachant la stratégie des autres agents. Il est intéressant de noter que dans le cas d'un jeu stochastique à un seul joueur (un MDP donc), la stratégie de l'équilibre de Nash correspond aux politiques optimales définies au chapitre précédent.

L'existence d'au moins un équilibre de Nash dans tous les jeux stochastiques est assurée par le théorème suivant (Filar and Vrieze (1996)) :

Théorème 4.2.1. *Dans un jeu stochastique G, il existe un équilibre de Nash pour des stratégies stationnaires.*

Deux remarques doivent être présentées ici. Tout d'abord, il peut exister plusieurs équilibres de Nash. Choisir l'équilibre le plus judicieux est important lorsque l'on travaille avec des jeux stochastiques, cependant un tel équilibre peut être difficile à définir. Deuxièmement, contrairement au cas des MDP, il se peut que l'équilibre de Nash ne soit pas en stratégies déterministes mais seulement stochastiques.

4.2.3 Jeu à somme nulle

Il existe deux façons de considérer un jeu stochastique à somme nulle : on peut tout d'abord voir deux agents visant à maximiser deux fonctions Q opposées mais on peut également considérer une seule fonction Q, dont le premier agent (appelé *maximiseur*) cherche à la maximiser et dont le second (le *minimiseur*) vise à la minimiser.

On peut prouver que si les deux joueurs suivent ces stratégies de maximisation et de minimisation, le jeu converge vers un équilibre de Nash étant le seul du jeu (Filar and Vrieze (1996)). Dans ce cas, grâce au théorème du Minmax (Shapley (1953)), la valeur du jeu est (avec le joueur 1 maximisant et le joueur 2 minimisant) :

$$\begin{aligned} V^* &= \max_{\sigma_1}\min_{\sigma_2} V(\sigma_1,\sigma_2) \\ &= \min_{\sigma_2}\max_{\sigma_1} V(\sigma_1,\sigma_2) \end{aligned}$$

Comme nous le verrons dans les sections suivantes, l'existence de cette unique fonction de valeur est très utile pour trouver des algorithmes efficaces de résolution des jeux stochastiques.

4.3 Algorithmes

L'apprentissage par renforcement dans les jeux stochastiques est un problème relativement jeune. À l'heure actuelle, l'existence d'un algorithme convergeant toujours de manière efficace vers un équilibre de Nash dans le cas où les agents apprennent grâce au même algorithme (procédure dite en *self-play*), est un problème ouvert. De

nombreux algorithmes ont cependant été proposés (Buşoniu et al. (2008)), chacun avec ses caractéristiques, forces et faiblesses.

Les premières techniques d'apprentissage par renforcement pour résoudre les jeux stochastiques ont été introduites dans Littman (1994). Dans son article est présenté minimax-Q, une variante de l'algorithme Q-Learning adaptée aux jeux stochastiques à somme nulle dans lequel l'opérateur de maximisation est remplacé par un opérateur minmax. Cet algorithme est garanti de toujours converger vers l'équilibre de Nash en self-play dans un jeu purement compétitif. Ce travail a ensuite été étendu dans Littman (2001) avec Friend-or-Foe Q-Learning (FFQ), un algorithme assuré de converger vers l'équilibre de Nash dans les environnements purement coopératifs ou purement concurrentiels. Une extension de ces travaux a par la suite été proposée par Hu and Wellman (2003), qui a été le premier algorithme adapté aux jeux stochastiques à somme générale. Cet algorithme, Nash-Q, est également une variante de Q-Learning capable de permettre aux agents d'atteindre un équilibre de Nash sous certaines conditions. Celles-ci sont cependant extrêmement restrictives sur la distribution des récompenses. Si ces hypothèses ne sont pas vérifiées, il est empiriquement prouvé que la convergence n'est plus garantie. Ce résultat est consolidé dans Zinkevich et al. (2006), qui prouve par un contre-exemple que la Q-fonction ne contient pas suffisamment d'informations pour permettre une convergence vers un équilibre de Nash dans le cadre des jeux à somme générale. Les algorithmes basés uniquement sur Q-Learning ne peuvent donc être à même de résoudre les jeux stochastiques.

Lorsque les paramètres du jeu stochastique sont connus, des méthodes basées sur la programmation linéaire multi-objectifs (Herings and Peeters (2000), Dermed and Isbell (2009)) permettent de trouver un équilibre de Nash, voir même de le choisir dans certains cas (grâce à l'algorithme de suivi stochastique). Ces méthodes sont cependant connues pour passer difficilement à l'échelle, et en raison de cela, dès que le jeu devient trop grand, elles ne deviennent plus applicables.

Akchurina (2009) propose un algorithme d'apprentissage basé sur la théorie des jeux évolutionnaire convergeant empiriquement vers un équilibre approximatif lorsque le jeu est connu. Le suivi de l'équilibre est effectué ici en résolvant à chaque itération un système d'équations différentielles ordinaires, ce qui pose rapidement des problèmes de calculabilité.

Les deux approches précédentes supposent que le jeu est connu (ou au moins qu'un modèle de celui-ci a été appris). A fortiori, ils supposent également que chaque agent récupère la valeur de la récompense affectée aux autres agents (cette hypothèse est également faite dans les algorithmes basés sur les Q-fonctions). Cette hypothèse peut être contraignante. Par exemple, dans le cas du dialogue et des interactions humain/machine, les objectifs de l'utilisateur sont inconnus de la machine. Pérolat et al. (2018) proposent un algorithme basé sur les architectures acteur-critique et sur les méthodes d'approximation stochastique à deux échelles de temps n'ayant pas besoin de faire cette hypothèse. Cet algorithme est un algorithme d'apprentissage en-ligne dont la convergence est garantie dans le cas d'un jeu à somme nulle et dans le cas d'un jeu purement coopératif.

Parmi tous ces algorithmes, tous ceux basés sur l'apprentissage par renforcement apprennent en ligne. Des algorithmes ont cependant été conçus pour traiter le cas hors-ligne. L'algorithme LSPI, abordé en Section 2.2.5.2, fut tout d'abord conçu pour traiter le cas des jeux stochastiques à somme nulle (Lagoudakis and Parr (2002)). Cet algorithme a ensuite été étendu dans Pérolat et al. (2016) afin d'améliorer ses performances et sa stabilité. Le cas des jeux à somme générale a, par la suite, été traité par Perolat et al. (2017). Cet algorithme, basé sur la minimisation hors-ligne du résidu de Bellman, est garanti de trouver un équilibre de Nash d'un jeu stochastique à somme générale. Cependant, il requiert un grand nombre de données pour pouvoir fonctionner et la sélection d'équilibre ne peut être réalisée.

Dans ce chapitre, nous utiliserons deux autres algorithmes détaillés dans les sections suivantes : WoLF-PHC (Bowling and Veloso (2002)) and AGPI-Q (Perolat et al. (2015)).

4.3.1 WoLF-PHC

WoLF-PHC est une extension de l'algorithme Q-learning permettant d'apprendre des stratégies probabilistes. Il considère que les agents indépendants évoluent dans un environnement rendu non stationnaire par la présence des autres. Dans un tel cadre, le but des agents n'est pas de trouver un équilibre de Nash (ce n'est donc pas un algorithme de jeux stochastiques), mais de faire le mieux possible dans un environnement rendu non-stationnaire par la présence d'autres agents apprenant (et par conséquent éventuellement converger vers un équilibre de Nash lorsque l'algorithme est utilisé en self-play). L'algorithme est basé sur l'idée suivante : la convergence doit être facilitée si les agents apprennent à s'adapter rapidement lorsqu'ils sont sous-optimaux et apprennent lentement lorsqu'ils sont presque optimaux (afin de permettre aux autres agents de s'adapter à cette stratégie).

WoLF-PHC met à jour les Q-valeurs comme dans Q-learning. La probabilité de sélectionner la meilleure action est augmentée de façon incrémentale selon un certain taux d'apprentissage (variable) δ, qui se décompose en deux taux d'apprentissage δ_L et δ_W, avec $\delta_L > \delta_W$. La mise à jour de la stratégie est faite selon δ_L pendant la perte et à δ_W en gagnant.

Pour déterminer si un agent perd ou gagne, la valeur attendue de sa stratégie réelle π est comparée à la valeur attendue de sa politique moyenne $\overline{\pi}$. Formellement, un agent gagne si $\sum_a \pi(s,a)Q(s,a) > \sum_a \overline{\pi}(s,a)Q(s,a)$ et perd autrement.

WoLF-PHC est formellement défini par l'Algorithme 8.

La convergence de l'algorithme est prouvée dans le cas 2 états 2 actions mais dans le cas général, celle-ci n'est pas prouvée. Il est même montré sur des exemples jouets que parfois, l'algorithme ne converge pas (Bowling and Veloso (2002)).

Algorithme 8 WoLF-PHC

Entrées : Une Q-fonction initiale quelconque, un paramètre d'apprentissage α, une politique π et deux paramètres δ_L et δ_W, un compteur $C(s) = 0$

Pour Chaque épisode **faire**

Tant que L'épisode n'est pas terminé **faire**

Choisir une action $a \in \mathcal{A}$ à exécuter

Récupérer l'état suivant s' et la récompense reçue $r(s, a)$

Mettre à jour $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big]$

Mettre à jour la politique moyenne $\overline{\pi}$

$C(s_t) \leftarrow C(s_t) + 1$

$\forall a \in \mathcal{A}$, $\overline{\pi}(s_t, a) \leftarrow \overline{\pi}(s_t, a') + \frac{1}{C(s_t)}(\overline{\pi}(s_t, a) - \pi(s_t, a))$

Mettre à jour $\pi(s_t, a)$ et la contraindre à être une distribution de probabilité

$$\pi(s_t, a) \leftarrow \pi(s_t, a) + \begin{cases} \delta \text{ si } a = \text{argmax}_{a'} Q(s_t, a') \\ \frac{-\delta}{|\mathcal{A}_i|-1} \text{ sinon} \end{cases}$$

$$\text{avec } \delta = \begin{cases} \delta_W \text{ si } \sum_a \pi(s_t, a)Q(s_t, a) > \sum_a \overline{\pi}(s_t, a)Q(s_t, a) \\ \delta_L \text{ sinon} \end{cases}$$

Fin tant que

Fin pour

4.3.2 AGPI-Q

Approximate Generalized Policy Iteration-Q, ou AGPI-Q (Perolat et al. (2015)), est une extension de l'algorithme Fitted-Q (Gordon (1999), Ernst et al. (2005), Riedmiller (2005)) résolvant les jeux stochastiques à somme nulle de manière hors-ligne. À l'étape d'initialisation, N échantillons (s, a_1, a_2, r, s') et une Q-fonction sont donnés. L'algorithme consiste alors en K itérations, chacune se décomposant en deux parties : une *étape gloutonne* et une *étape d'évaluation*. À chaque itération, une meilleure estimation de la Q-fonction optimale est alors donnée.

Soit $\mathcal{D} = (s^j, a^j, b^j, r^j, s'^j)$ un ensemble de N transitions. Au pas de temps $k + 1$, l'*étape gloutonne* trouve la meilleure action du maximiseur $\overline{a}$ du jeu matriciel défini par $Q_k^j(s'^j, a^j, b^j)$. Dans notre cas, où les agents agissent de manière alternée, cette étape consiste à trouver un maximum. Puis, lors de l'étape d'évaluation considérant que le minimiseur joue sa stratégie minimisante, la Q-valeur suivante est alors calculée $Q^j = r + \gamma \min_b Q_k^j(s'^j, \overline{a}^j, b)$. Enfin, la Q-fonction est alors régressée dans un espace d'hypothèses donné. L'algorithme complet est décrit en Figure 9

4.4 Le dialogue comme un jeu stochastique

Le dialogue est une interaction entre plusieurs agents. Grâce aux jeux stochastiques, il est envisageable de le considérer comme tel lors du processus d'optimisation. Si chaque agent (*c'est-à-dire* l'utilisateur et le DM) a ses propres objectifs et prend ses décisions pour les atteindre, il semble naturel de modéliser chacun d'eux en tant que MDP. Cependant, dans le cadre traditionnel, cela ne se fait que sur l'un des deux

Algorithme 9 AGPI-Q

Entrées : Un ensemble de transitions $\mathcal{D} = \{(s_i, a_i^1, a_i^2, s'_i, r_i)\}_{i=1}^N$, $q_0 = 0$ une Q-fonction, $\mathcal{F}$ un espace d'hypothèses
Pour $k = 1..K$ **faire**
  Pour $i = 1..N$ **faire**
    Résoudre $\overline{a}_i = \arg\max_a \min_b \hat{Q}_k(s_i, a, b)$
  Fin pour
  Pour $i = 1..N$ **faire**
    Mettre à jour $q_{i,k} = r_i + \gamma \min_{b \in \mathcal{A}^2} \hat{Q}_k(s'_i, \overline{a}, b)$
  Fin pour
  Résoudre le problème d'apprentissage supervisé $\hat{Q}_k = \text{argmin}_{q \in \mathcal{F}} \sum_{i=1}^N l(q_{i,k}, q)$ où l est une fonction de coût
Fin pour
Retourner $\pi_K(s) \in \text{argmax}_a \hat{Q}_K(s, a)$, $\forall s \in \mathcal{S}$

interlocuteurs. Depuis Levin and Pieraccini (1997), Singh et al. (1999), seul le DM est encodé en tant qu'agent RL, et ce malgré de rares exceptions (Chandramohan et al. (2011, 2012b,a)). L'utilisateur est plutôt considéré comme un agent stationnaire pouvant être modélisé par exemple comme un réseau bayésien (Pietquin (2006)), comme un processus basé sur un agenda (Schatzmann et al. (2007)), ce qui conduit à des erreurs de modélisation (Schatztnann et al. (2005), Pietquin and Hastie (2013)) comme abordé en Section 3.2.4.1.

À première vue, il semble raisonnable de penser que si deux agents RL, précédemment formés pour atteindre une stratégie optimale, interagissent les uns avec les autres, cela générerait des dialogues optimaux. Cette affirmation cependant est fausse. Chaque agent serait optimal compte tenu de l'environnement sur lequel il a été formé, mais dans un autre environnement, on ne peut rien dire sur la politique apprise. En outre, si deux DM sont formés avec des techniques RL traditionnelles, aucune convergence n'est garantie puisque, comme on l'a vu plus haut, des non-stationnarités apparaissent. En effet, la non-stationnarité n'est pas bien gérée par les méthodes d'apprentissage par renforcement standards, même si certaines méthodes pourrait y faire face comme Geist et al. (2009), Daubigney et al. (2012), l'adaptation ne serait a priori pas assez rapide.

Optimiser de manière jointe des agents apprenant par renforcement selon le cadre des jeux stochastiques permet de trouver un équilibre de Nash. Cela permettrait alors de garantir que les deux stratégies obtenues sont optimales l'une par rapport à l'autre. De cette particularité provient la différence fondamentale avec les travaux précédents Chandramohan et al. (2012b), Georgila et al. (2014), Efstathiou and Lemon (2014).

Dans la section suivante, nous illustrons comment le dialogue peut être modélisé par un jeu stochastique, et comment les fonctions de transition et de récompense dépendent de la stratégie des deux agents. Nous proposons pour cela un jeu à somme nulle dans lequel les agents doivent récupérer une information plus rapidement que leur interlocuteur. Ici, un utilisateur humain (l'agent 1) et le DM (l'agent 2) sont modélisés par des MDPs : chacun a ses objectifs encodés dans une fonction de récompense $\mathcal{R}_1$ et

$\mathcal{R}_2$ qui dépend de l'action jointe.

4.5 Un jeu de dialogue à somme nulle

Nous avons choisi de valider notre approche sur une preuve de concept, une simulation de dialogue orienté tâche. La tâche considérée ici implique deux agents, chacun d'entre eux reçoit un numéro secret aléatoire et cherche à deviner le numéro de l'autre agent. Ces agents sont adversaires : si l'un gagne, l'autre perd.

Pour trouver le nombre secret de son interlocuteur, les agents ont les actions suivantes à leur disposition : `demander`, `répondre`, `deviner`, `ok`, `confirmer` et `écouter`.

Pour récupérer des informations sur le numéro caché de son adversaire, un agent peut `demander` si ce nombre est plus petit ou plus grand qu'un autre nombre. Son adversaire est alors contraint de `répondre` la vérité. Pour montrer qu'il a compris la réponse, l'agent dit `ok` et redonne le tour à son adversaire, qui cherchera alors à deviner le nombre.

Ces agents ne sont pas parfaits, ils peuvent mal comprendre ce qui a été dit. Cette caractéristique a été ajoutée afin de simuler les erreurs de l'ASR et du NLU dans les SDS réels. Ils sont un indicateur donnant un indice sur la probabilité d'avoir bien compris (un niveau de confiance). Ils ne sont donc jamais certains de ce qu'ils ont entendu et ils peuvent répondre à une mauvaise question, par exemple dans l'échange suivant :

- Est ce que ton nombre secret est supérieur à x ?

- Mon nombre est supérieur à y.

Lorsqu'une telle erreur survient, l'Agent 1 est autorisé à poser une autre question plutôt que de simplement dire `ok`. Cette punition est sévère pour l'agent qui a mal compris, c'est presque comme s'il devait passer son tour. Un autre acte de dialogue est proposé pour faire face à de telles situations. Si un agent n'est pas sûr, il peut demander à `confirmer`. Dans ce cas, l'agent 1 peut poser sa question à nouveau. Pour éviter les abus, *i.e.* demande indéfiniment une confirmation, cette action induit un coût (et donc un gain pour l'adversaire).

Si un agent pense avoir trouver le nombre exact de son adversaire, il peut choisir de `deviner`. Si sa réponse est exacte, il gagne. Dans le cas contraire, il perd. Un exemple de dialogue est présenté en Figure 4.2.

Dans la mesure où nous modélisons le dialogue comme une interaction au tour par tour et que nous devons considérer des actions jointes, l'action `écouter` (correspondant à l'action nulle) est introduite.

4.5.1 Cadre expérimental

Les effets du cadre multi-agents sont étudiés ici grâce à une caractéristique particulière du dialogue humain/machine : la gestion de l'incertitude due aux dysfonctionnements de l'ASR et du NLU. Pour promouvoir des algorithmes simples, nous avons effectué nos expériences sur le jeu de dialogue à somme nulle présenté ci-dessus.

DM	Est ce que ton nombre secret est supérieur à 5 ?
Humain	Mon nombre est inférieur à 5
DM	Ok
Humain	Est ce que ton nombre secret est supérieur à 5 ?
DM	Mon nombre est supérieur à 8
Humain	Est ce que ton nombre secret est supérieur à 9 ?
DM	Que m'as tu demandé ?
Humain	Est ce que ton nombre secret est supérieur à 9 ?
DM	Mon nombre est supérieur à 9
Humain	Ok
DM	Est ce que ton nombre secret est supérieur à 3 ?
Humain	Mon nombre est inférieur à 3
DM	Ok
Humain	Est ce que ton nombre secret est 10 ?
DM	Oui

FIGURE 4.2 – Exemple de dialogue correspondant à ce jeu. L'erreur de compréhension du DM (en rouge) lui fait perdre le jeu

Sur cette tâche, nous comparons trois algorithmes : Q-Learning, WoLF-PHC et AGPI-Q. Parmi ces algorithmes, seul AGPI- Q converge de manière prouvée vers un équilibre de Nash dans un cadre multi-agent. Q-Learning et WoLF-PHC ont cependant tous les deux été utilisés comme algorithmes d'apprentissage Multi-Agent dans un cadre de dialogue (par respectivement English and Heeman (2005) et Georgila et al. (2014)). À l'instar de ces travaux, les expériences sont réalisées en simulation. Nous montrerons que, contrairement à AGPI-Q, Q-Learning et WoLF-PHC ne convergent pas vers l'équilibre de Nash et ne sont donc pas nécessairement judicieux dans le cadre du dialogue.

4.5.1.1 Un modèle d'erreur pour l'ASR et le SLU

Une difficulté rencontrée lorsque l'on travaille avec les systèmes de dialogue est la manière dont le DM fait face à l'incertitude résultant des erreurs d'ASR et de NLU, à l'aide de leurs scores de confiance. Bien que Georgila et al. (2014) ne modélisent pas ces erreurs, nous avons choisi de les représenter dans notre tâche car celles-ci peuvent être sources de difficultés.

Les scores renvoyés par les ASR et SLU ne correspondent pas toujours une probabilité. La seule hypothèse faite ici est que avec un score inférieur (resp. supérieur) à 0,5, la probabilité de mal comprendre la dernière énonciation est plus grande (resp. plus petite) que 0,5. Comme les dialogues sont simulés, ces niveaux de confiance ASR et NLU sont également simulés. On choisit ici le modèle d'ASR proposé dans Khouzaimi et al. (2015).

Chaque agent possède un certain taux d'erreur phrase constant SER_i (en anglais

Sentence Error Rate). Chaque énoncé u_t est reçu correctement compris par l'agent i avec une probabilité $(1 - \text{SER}_i)$. Autrement, cet énoncé est remplacé par un autre.

Le score de confiance est ensuite modélisé de la façon suivante. Un score $(-\infty, \infty)$ est ensuite échantillonné selon une distribution normale centrée en -1 lors d'une compréhension incorrecte et $+1$ lors d'une compréhension correcte. Un score compris dans l'intervalle (0,1) est obtenu en appliquant la fonction sigmoïde $f(x) = \frac{1}{1+\exp(-x)}$, sur le précédent score compris dans l'intervalle $(-\infty, \infty)$.

Puisque Q-Learning et WoLF-PHC sont utilisés sur cette tâche sous leur forme tabulaire, ce score de confiance a été discrétisé. Pour avoir des états où l'agent est presque sûr d'avoir compris (ou sûr d'avoir mal compris), cette discrétisation a été appliquée en partageant le score autour des points de coupe 0,1, 0,5 et 0,9. Par souci d'équité, cette discrétisation a été appliquée pour l'algorithme AGPI-Q.

4.5.2 Modélisation de la tache

4.5.2.1 Espace d'états

Considérons deux agents i et j. Leurs nombres secrets sont respectivement m et n. Pour collecter suffisamment d'information sur m, l'agent i demande à l'agent j si son nombre secret m est inférieur ou supérieur à un nombre donné k. Si l'agent j répond que m est plus grand (respectivement plus petit) que k, l'agent i retient que k est une borne inférieure b_i (respectivement une borne supérieure b'_i) de m. Les connaissances de l'agent i sur m peuvent donc être représentées par l'intervalle $I_i = [b_i, b'_i]$. La probabilité de gagner en `devinant` est alors donnée par $p = \frac{1}{b'_i - b_i + 1}$. Le progrès de l'agent i dans le jeu peut donc mesurer à partir de la valeur $c_i = b'_i - b_i + 1$, le cardinal de I_i. Au début du jeu, on a : $I_i = I_j = [1, 5]$. Étant donné que les agents doivent connaître la progression de l'ensemble, ils suivent tous deux c_i et c_j.

Pour prendre une action cohérente, un agent doit se rappeler qui a prononcé le dernier énoncé, quel a été le dernier énoncé qu'il a entendu et dans quelle mesure il croit que ce qu'il a entendu était ce qui avait été dit.

En résumé, les agents prennent leur décision selon les caractéristiques suivantes : le dernier énoncé, sa confiance dans cet énoncé, qui l'a prononcé, son progrès dans le jeu et les progrès de son adversaire. Ils n'ont donc pas besoin de suivre l'ensemble des nombres secrets possibles, mais seulement le cardinal de ces ensembles. L'espace d'état contient donc des $2 * 5 * 4 * 5 * 5 = 1000$ états.

4.5.2.2 Espace d'actions

Les agents ont à leur disposition les actions suivantes : `demander`, `répondre`, `deviner`, `confirmer` et `écouter`. Les actions `demander`, `répondre` et `deviner` nécessitent un argument : le nombre auquel l'agent décide de se comparer. Afin d'apprendre plus vite, nous avons choisi de ne pas apprendre cette valeur. Lorsqu'un agent utilise l'action `demander`, il demande si le nombre secret de son adversaire est plus grand ou plus petit que le nombre au milieu de son intervalle de confiance (cet intervalle est

calculé par l'environnement, il n'est pas pris en compte dans l'état). En répondant, l'agent répond directement si son nombre secret est plus grand ou plus petit que le nombre entendu (qui n'est donc pas nécessairement le nombre proposé par son adversaire). L'action `deviner` permet à l'agent de deviner un nombre tiré aléatoirement parmi l'ensemble compris des nombres possibles.

4.5.2.3 Fonction de récompense

Dans la mesure où les fonctions de récompense sont opposées, pour caractériser le jeu, il suffit de considérer le joueur maximiseur. Supposons que ce soit son tour. Si il devine correctement le nombre secret de son adversaire, il reçoit la récompense $+1$. Si il demande une confirmation, il reçoit -0.2. Il n'a donc jamais intérêt à bloquer le dialogue en demandant indéfiniment (dans ce cas en effet, l'agent reçoit une récompense de $-0.2\sum_{k=0}^{\infty}(\gamma^2)^k \approx -1.05$ pour $\gamma = 0.9$ tandis que si il perd, sa récompense, de -1, est supérieure). Pour toutes les autres actions, l'agent ne reçoit aucune récompense.

4.5.3 Entraînement des algorithmes

Pour que la comparaison avec le travail réalisé dans Georgila et al. (2014) soit la plus juste possible, nous avons choisi de suivre leur méthode d'entraînement des agents apprenant avec Q-Learning et avec WoLF-PHC. Ces deux algorithmes sont entraînés en *self-play* tout en suivant une stratégie ϵ-gloutonne. L'entraînement est partagé en cinq passes (*epochs*) de 100 000 dialogues. On utilise pour le paramètre d'exploration ϵ des valeurs de 0.95 pour la première passe, 0.8 pour la seconde, 0.5 pour la troisième, 0.3 pour la quatrième et 0.1 pour la cinquième, résultant en une grosse exploration au début de l'entraînement puis une exploration moins agressive par la suite.

Les paramètres δ_L et δ_W de l'algorithme WoLF-PHC ont été fixés à $\delta_W = 0.05$ et $\delta_L = 0.2$. Le rapport $\delta_L/\delta_W = 4$ assure donc un apprentissage agressif lorsque l'agent estime qu'il perd.

AGPI-Q, en tant qu'algorithme d'apprentissage par renforcement hors-ligne, requiert un ensemble de données pour apprendre. Afin de les générer, nous avons suivi la méthode proposée dans Pietquin et al. (2011a). Une politique optimale (ou quasi-optimale au moins) est tout d'abord implémentée à base de règles. Dans notre cas, cette politique est la suivante : un agent demande toujours à obtenir plus d'informations à propos du nombre secret de son adversaire à moins que l'un des deux agents ait suffisamment d'information pour pouvoir deviner ce nombre avec une probabilité 1. Lorsque l'agent doit répondre, il demande une confirmation si son score de confiance est strictement inférieur à 0.5.

Cette politique est ensuite rendue stochastique afin de permettre une exploration. Les agents prennent leurs décisions en fonction de la politique implémentée à base de règles avec une probabilité ϵ et choisissent aléatoirement une autre action avec une probabilité $(1 - \epsilon)$. Les échantillons $\langle s, a_1, a_2, r, s' \rangle$ sont ensuite collectés. Nous nous assurons ainsi que le problème est suffisamment bien exploré et qu'il existe des échantillons menant à un succès dans la réalisation de la tâche (et assignant ainsi

la récompense maximum +1). Afin d'assurer la convergence de l'algorithme, 75 000 dialogues sont ainsi collectés.

Afin que le modèle ne dépende que d'un minimum de paramètres, les arbres CART (Breiman (1984)) sont utilisés en tant qu'espace d'hypothèses pour la régression.

Chaque algorithme est entraîné avec un DM possédant un taux d'erreur phrase de 0, 0.1, 0.2, 0.3 et 0.4.

4.5.4 Résultats

Les points de décision dans ce jeu sont au nombre de deux : quel est le meilleur moment pour terminer le dialogue à l'aide de l'action `deviner` et quel est la meilleure manière de gérer l'incertain par le biais de l'action `confirmer`. Nous avons choisi de caractériser la convergence vers l'équilibre de Nash en représentant la durée moyenne d'un dialogue ainsi qu'en représentant le nombre de fois que l'action `confirmer` est utilisée. Ces deux résultats sont calculés en moyennant sur 5000 dialogues. Les Figures 5.5 et 4.4 illustrent ces résultats.

Nous observons tout d'abord sur la première figure que la longueur des dialogues générés avec l'algorithme Q-Learning diminue lorsque le taux d'erreur mot augmente. Ceci est expliqué par la Figure 4.4 : les agents ayant appris par Q-Learning n'apprennent pas à utiliser l'action `confirmer`. Par ailleurs, la longueur de ces dialogues n'est pas régulière, ce qui prouve que les algorithmes ne convergent pas vers une politique stable. Ces deux résultats sont ainsi conformes à la théorie : Q-Learning est un algorithme lent et c'est pourquoi les agents n'ont pas suffisamment de temps pour s'adapter aux non-stationnarités provenant de l'environnement multi-agents. La convergence n'est donc pas possible.

Nous observons que WoLF-PHC ne gère pas non plus correctement l'incertitude. Nous observons sur la Figure 4.4 que, par dialogue, si le nombre de fois que l'action `confirmer` est utilisée est de loin le plus grand, il reste constant. Ainsi, si le SDS demande toujours une confirmation, y compris lorsqu'il n'y a pas de bruit, c'est vraisemblablement car il est constamment en situation de désavantage, il perd tout le temps. Ainsi, en perdant, son taux d'apprentissage lui fait constamment changer de stratégie, au dépend d'une stabilité d'apprentissage qui lui aurait permis d'être plus efficace. Aussi, comme évoqué précédemment, dans un contexte multi-agent, la convergence de cet algorithme n'est pas garantie.

Nous observons alors que AGPI-Q est le seul des trois algorithmes étant robuste contre le bruit. Nous observons que la longueur des dialogues ainsi que le nombre d'actions `confirmer` augmentent progressivement avec une augmentation du taux d'erreur phrase du système de dialogue. Nous sommes par ailleurs assurés par la théorie que dans ce cadre, aucune amélioration n'est possible.

Il est finalement intéressant que des stratégies non-triviales puissent émerger de l'interaction entre les agents AGPI-Q. Par exemple, lorsque les deux agents sont quasiment à la fin d'un dialogue ($c_i = 2$ pour chaque agent), ceux-ci choisissent de tenter de deviner, quand bien même ils n'ont que très peu de chances de gagner. Cette décision

est également prise lorsqu'ils sont sûrs que leur adversaire gagnera au tour prochain.

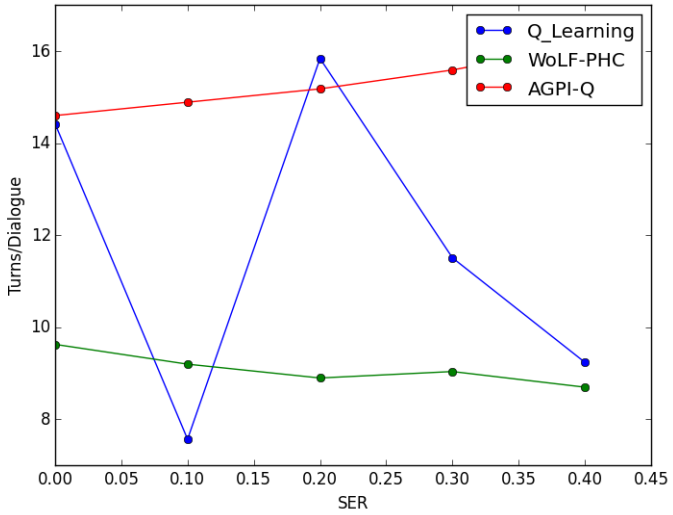


FIGURE 4.3 – Longueur des dialogues

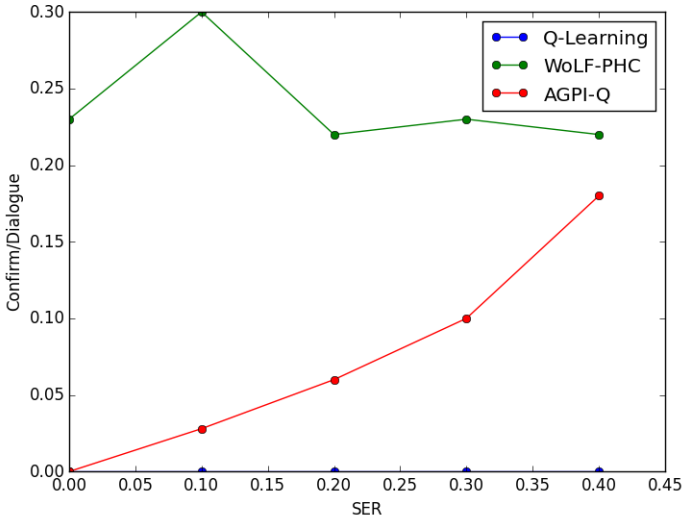


FIGURE 4.4 – Fréquence d'utilisation de l'action `CONFIRMER`

4.6 Conclusion

Nous avons au cours de ce chapitre élaboré un modèle de co-apprentissage entre un humain et son système de dialogue. Ainsi, les phénomènes de non-stationnarités dus à

l'adaptation de l'humain à son système sont prises en compte. Par ailleurs, nous avons vu que ce cadre permettait d'aller par delà les dialogues orientés tâches et permettait d'aborder de nouveaux types de dialogues dans lesquels l'être humain et la machine ne collaborent pas nécessairement. Notre modèle présente l'avantage d'être à la fois rigoureux d'un point de vue théorique et de permettre d'optimiser la politique des deux interlocuteurs.

Ce modèle présente l'être humain comme un agent apprenant et rationnel (*i.e.* cherchant à optimiser sa fonction de récompense). Nous allons voir au chapitre suivant un second cadre dans lequel l'humain n'est pas modélisé comme une partie désintéressée de l'environnement. Ici, l'être humain cherchera a accélérer l'apprentissage de son système de dialogue.

Chapitre 5

Incorporations de conseils humains pour l'apprentissage des systèmes de dialogue

Au chapitre précédent, nous avons présenté un modèle de dialogue humain/machine permettant la prise en compte de phénomènes de co-adaptation entre les différents interlocuteurs. La principale caractéristique de ce modèle est de voir l'humain non pas comme une distribution de probabilité stationnaire, mais comme un processus de décision markovien agissant de manière à réaliser ses objectifs. L'interaction dialogique en tant que telle n'est cependant pas la seule manière pour l'humain d'agir sur son système de dialogue pour que celui-ci se comporte de manière conforme à ses attentes. Dans ce chapitre, nous considérons un nouveau cadre d'interaction entre l'humain et la machine. Ici, l'humain se comporte comme un guide destiné à accélérer le processus d'optimisation de son système et de fait, il ne peut être modélisé comme simplement une partie de l'environnement.

5.1 Motivation

Nous avons montré au Chapitre 3 les avantages des méthodes statistiques pour l'optimisation de systèmes de dialogue. Plus particulièrement, aux cours des dernières années, la combinaison de l'apprentissage par renforcement avec l'apprentissage profond a permis de grandes améliorations dans les capacités de ceux-ci, que ce soit dans l'augmentation de leurs domaines d'application ou dans leurs performances (Fatemi et al. (2016), Strub et al. (2017)). Apprenant à partir de données brutes seulement, ces modèles sont attrayants en raison de leur efficacité et de leur généralité. Ainsi, une architecture peut traiter une grande variété de tâches sans nécessiter d'intervention humaine. Cette approche purement statistique se heurte cependant à deux majeures difficultés.

Tout d'abord, comme abordé en section 3.2.4, une grande difficulté dans l'optimisation statistique de systèmes de dialogue est le manque de données. Celles ci sont en effet chères et difficiles à collecter. Un système de dialogue se doit donc d'être performant malgré un nombre relativement faible de données sur lesquelles il a été entraîné. Les algorithmes d'apprentissage profond par exemple, bien qu'extrêmement performants, sont connus pour nécessiter un grand nombre de données pour être efficaces. Comme nous avons pu le voir en section 3.2.4 deux méthodes sont applicables pour gérer ce

manque de données. Dans un premier cas, les données collectées sont utilisées pour réaliser un simulateur d'utilisateur (Schatzmann et al. (2006), El Asri et al. (2016a)). Une politique de dialogue est apprise en ligne sur ce simulateur avant d'être affinée en étant appliquée sur de véritables êtres humains. Dans un second cas, des algorithmes d'apprentissage par renforcement hors-ligne sont utilisés directement sur les données collectées. La politique apprise peut alors être utilisée telle quelle (et éventuellement optimisée en ligne) sur des êtres humains.

La deuxième limitation est plus subtile et peu sembler à l'opposé de la philosophie de l'approche *end-to-end* : dans un système purement statistique, aucune expertise humaine n'est a priori facilement injectable durant le processus d'optimisation. Cette expertise est cependant nécessaire dans le cas d'un système de dialogue. Dans le cas de la diffusion industrielle de ces systèmes de dialogue, il est nécessaire d'assurer des comportements sûrs et de limiter (voir interdire) certaines actions non désirables. En effet dans ce cas, une mauvaise expérience utilisateur, même uniquement lors de la sortie du produit, peut négativement affecter son expansion. Ces comportements indésirables peuvent être de différents types.

Une classification des contraintes de sécurités relatives aux systèmes de dialogue est donnés dans Henderson et al. (2017). Tout d'abord, il est nécessaire d'assurer que les énoncés du systèmes de dialogue soient linguistiquement corrects et le moins ambigus possible. Ensuite, dans certains domaines d'application tels que l'assistance médicale, les performances du système de dialogue doivent être garanties pour ne pas donner lieu à un faux diagnostic et pouvant alors entraîner de graves dommages collatéraux. Enfin, les énoncés du système de dialogue se doivent de respecter de bonnes manières afin de ne pas heurter la sensibilité de son interlocuteur, notamment dans des cas critiques comme dans l'exemple précédent d'aide au diagnostic médical. Assurer cette sécurité dans les comportements est un problème difficile si l'on part seulement d'un jeu de données. Généralement, la supervision humaine est nécessaire.

De nombreux travaux se sont concentrés sur l'intégration d'un guidage humain pour les système de dialogue parlé basés sur l'apprentissage par renforcement (Singh et al. (2002), Williams (2008), Laroche et al. (2010), Chen et al. (2017)) ou plus généralement tout agent apprenant par renforcement (Maclin and Shavlik (1996)). Dans la plupart des cas, ces travaux résolvent le problème en restreignant les actions disponibles au système dans certains états spécifiés préalablement par un humain. Ils peuvent également suivre certaines règles dans ces états. La machine ne peut alors pas son améliorer son comportement, elle ne fait qu'obéir aux ordres de l'être humain. Une fois que des comportements robustes ont été appris grâce à l'apprentissage par renforcement, ils peuvent être remplacés petit à petit par des comportements appris. Cependant, une fois que les règles sont remplacées par la politique apprise, aucun contrôle n'est permis.

Une alternative à cette approche est fournie par le *Reward Shaping* (Ng et al. (1999)) et le cadre TAMER+RL (Knox and Stone (2010)). Ces deux cadres abordent le problème en encodant la supervision sous la forme d'une fonction de récompense additionnelle. Ainsi dans ces cadres, le guidage humain peut être vu comme une série de suggestions ou conseils donnés à la machine. Si un comportement semble béné-

fique, alors l'humain peut l'encourager. Dans le cas contraire, il peut être découragé. Le *Reward Shaping* agit en biaisant la fonction de récompense lors de l'apprentissage (il biaise donc également les Q-valeurs apprises) pour correspondre aux recommandations des experts. Le cadre TAMER+RL permet quant à lui d'agir sur la politique de l'agent apprenant. Contrairement aux méthodes exposées au paragraphe précédent, ces deux méthodes présentent l'avantage de fournir une garantie de convergence vers la solution du problème initial. Ils sont ainsi en mesure de surmonter certains biais humains qui conduiraient à des comportements sous-optimaux (Randlov and Alstrom (1998)). Cependant, contrairement à TAMER+RL, le Reward Shaping impose de faire des hypothèses fortes sur la forme de la fonction de récompense supplémentaire pour garantir cette invariance de la politique finale optimale. Trouver cette bonne fonction de récompense, incitant à agir selon les recommandations de l'expert et vérifiant ces hypothèses structurelles est malheureusement un problème difficile.

Toutes les méthodes précédentes présentent également l'inconvénient d'être conçues pour fonctionner dans le cadre de l'apprentissage par renforcement en ligne. Dans ce chapitre, nous proposons une méthode d'incorporation de conseils humains dans l'apprentissage par renforcement hors-ligne en introduisant une méthode basée sur le cadre TAMER+RL et sur l'algorithme LSPI. Cette méthode permet donc de combiner les avantages d'efficacité et de stabilité de l'apprentissage par renforcement hors-ligne avec la sécurité fournie par une supervision humaine.

La Section 5.2 fournit un état de l'art sur la conception d'agents apprenant par renforcement et étant capables de prendre en compte des conseils humains. Dans la section suivante, nous introduisons TAMER-LSPI, notre méthode pour intégrer des conseils humains sur une tâche d'Apprentissage par Renforcement Batch. Enfin, l'efficacité de notre approche est décrite dans la Section 5.3 sur un protocole expérimental basé sur une tâche simulée de réservation de restaurant conçue à partir de l'ensemble de données DSTC2.

5.2 Conseiller des agents

5.2.1 État de l'art

L'intégration de conseils experts accélérant le processus d'apprentissage pour des agents apprenant par renforcement et/ou fournissant des comportements sûrs est un problème ayant été abordé depuis le début des années 1990. Les premières solutions consistaient à oublier ce qui avait été appris par renforcement dans certains états et à appliquer une action choisie par un expert (Clouse and Utgoff (1992)) ou alors à dériver cette action d' un ensemble de règles, comme par exemple l'algorithme RATLE (Maclin and Shavlik (1996)). Dans cette ligne de travail, Laroche et al. (2010), Putois et al. (2010) proposent une modification de ce processus en concevant tout d'abord une stratégie sûre. Cette stratégie est ensuite affinée en ligne avec de l'apprentissage par renforcement, tout en ne laissant à chaque fois qu'un ensemble d'actions sûres à la disposition du système de dialogue.

Une variante de cette idée de ne proposer qu'un ensemble d'actions sûres à un système apprenant par renforcement a été proposée dans Williams (2008). Dans ce travail, les auteurs suggèrent d'utiliser tout d'abord un mécanisme apprenant à sélectionner un ensemble d'actions sûres grâce à un modèle dont les paramètres ont été précautionneusement choisis par un expert humain, puis de sélectionner l'énoncé du système parmi ces actions sûres par renforcement.

Ces méthodes présentent cependant plusieurs inconvénients. Tout d'abord se pose le problème de la généralisation. Choisir les états critiques dans lesquels l'agent serait susceptible d'avoir une action désastreuse, puis implémenter une réponse adéquate dans cette situation demande beaucoup de travail. Généralement, dans un dialogue, ces états critiques sont nombreux, il est donc nécessaire d'avoir une méthode permettant de généraliser ces conseils à des situations non annotées par l'expert humain. L'emploi de règles limite fortement ce problème de généralisation. Par ailleurs, une autre difficulté est de choisir le moment où il est possible de s'affranchir des conseils humains pour appliquer la politique apprise. Cette sélection est difficile car elle demande de constamment surveiller le comportement du système de dialogue pour savoir si celui-ci est suffisamment sûr. Les méthodes que nous allons exposer permettent de pallier ces deux problèmes. Tout d'abord des conseils estimés sont pris en compte dans les états non annotés par les experts. L'impact de ces conseils décroît au fur et à mesure du temps pour converger vers la solution du problème initial.

Le *Reward Shaping* (Ng et al. (1999)) fournit une alternative pour incorporer des conseils avec une approche plus subtile. Dans ce cadre, une récompense supplémentaire F conçue par un être humain est ajoutée à la récompense traditionnelle r. L'idée derrière cette deuxième récompense est que, parfois, la propagation du signal de récompense à travers l'ensemble du MDP est difficile et peut nécessiter un grand nombre d'échantillonnages pour séparer le signal du bruit. Ainsi, l'agent peut avoir des difficultés à distinguer quelles actions mènent au succès de la tâche ou à son échec. Pour surmonter cela, la récompense conçue par l'humain encourage explicitement l'agent à prendre de bonnes actions (ou estimées comme telles par l'être humain) et à éviter les mauvaises. Ng et al. (1999) prouvent cependant qu'une condition nécessaire et suffisante pour avoir une politique optimale finale identique à la politique optimale dans le MDP initial (*i.e.* sans la récompense additionnelle), est que la récompense additionnelle doit dériver de potentiels d'état, : $F(s, s') = \gamma\Phi(s') - \Phi(s)$. Wiewiora et al. (2003) étendent ce résultat en montrant qu'il était également possible de concevoir une fonction de mise en forme à partir de potentiels de couples état/action : $F(s, a, s', a') = \gamma\Phi(s', a') - \Phi(s, a)$. Dériver la fonction de mise en forme des potentiels est cependant une forte limitation du *Reward Shaping*. Un agent apprenant par renforcement prend en effet ses actions par rapport à la fonction Q. La récompense additionnelle doit donc respecter la propriété des potentiels tout en conduisant lors de l'apprentissage à une fonction Q correspondant aux conseils humains. Harutyunyan et al. (2015) montrent que concevoir une telle fonction conduit à un problème aussi difficile que celui de l'apprentissage par renforcement.

Ce paradigme a toutefois été appliqué avec succès dans une variété de problèmes

de dialogue. Par exemple El Asri et al. (2013) accélèrent la phase d'apprentissage d'un gestionnaire de dialogue en utilisant les retours comme potentiels, tandis que Su et al. (2015) utilisent des retours provenant d'un simulateur de dialogue pour accélérer l'apprentissage en interagissant avec de vrais utilisateurs. De la même manière, Ferreira and Lefevre (2015) montrent, sur une tâche en simulation, que faire du Reward Shaping à partir d'une récompense additionnelle basée sur des signaux sociaux non-vocaux, manifestant par exemple l'approbation ou la désapprobation de l'interlocuteur, permettait d'accélérer l'apprentissage d'un système de dialogue apprenant en ligne. Ce résultat a par la suite été validé sur des utilisateurs réels (Riou et al. (2018)).

Une approche alternative pour incorporer des conseils lors de l'optimisation d'un agent apprenant par renforcement est donnée par le Policy Shaping (Griffith et al. (2013)). Dans ce travail, les auteurs apprennent d'abord un modèle de la politique de conseil humain. Ce modèle est ensuite utilisé pour prédire quelle devrait être son action dans la situation de l'agent. Finalement, cette politique de conseil estimée est mélangée avec la politique apprise afin de dériver une politique finale tenant compte de ces conseils. Dériver la politique de conseils de l'expert est cependant un problème difficile. Pour résoudre ce problème, il faut se limiter à des modèles de conseils humains très basiques.

Dans le cas d'un agent apprenant en ligne par renforcement, biaiser la politique pour la faire correspondre aux conseils experts peut également être abordé sous l'angle de la gestion de l'exploration. Ainsi, Ferreira and Lefèvre (2013) proposent, à partir d'une politique stochastique, d'augmenter les probabilités de choisir les actions sélectionnées par l'expert, ce qui permet à l'agent apprenant d'explorer rapidement des zones intéressantes de l'espace d'états.

Récemment, Chen et al. (2017) ont introduit une méthode permettant d'incorporer des démonstrations directes d'experts pour orienter l'apprentissage de la politique vers une politique humaine sûre et efficace. Dans ce travail, les auteurs suggèrent de choisir l'action fournie par l'expert quand elle est disponible. Sinon, en introduisant un signal de récompense supplémentaire, ils incitent le système à prendre les actions que l'expert aurait probablement prises. Cette méthode souffre cependant du fait que l'agent apprenant peut ne pas être capable de surmonter les préjugés humains. En outre, déterminer ce qui peut être l'action humaine sur des états inconnus est une tâche difficile (Ross et al. (2011)).

Le problème de l'apprentissage par renforcement avec démonstrations expertes (Kim et al. (2013), Piot et al. (2014a), Hester et al. (2017b)) aborde également la tâche consistant à ajouter des informations expertes au signal de récompense traditionnel. Dans ce cas, ces informations supplémentaires sont des démonstrations d'actions optimales (ou quasi-optimales) dans certains états. Ce problème est généralement résolu en contraignant les Q-valeurs apprises à avoir des valeurs plus élevées dans les échantillons où une action experte a été fournie. Ce cadre ne peut cependant pas être appliqué dans notre cas car, par définition, les démonstrations des experts sont optimales et il n'est pas possible d'intégrer des retours négatifs et décourager les comportements non sûres. Ainsi dans ces cas, il est impossible de dire à l'agent qu'une action n'est pas

particulièrement bonne.

Enfin, une approche alternative de l'intégration des conseils humains pour les agents d'apprentissage par renforcement se concentre sur les façons de demander de l'aide à un expert humain lorsque l'agent rencontre des situations spécifiques où il n'a pas assez d'informations pour agir correctement. Ce problème a d'abord été traité dans Clouse (1996) et une première solution a été apportée avec le système `AskForHelp`, où l'agent apprenant par renforcement demande de l'aide quand toutes les valeurs Q dans un état sont trop similaires. Suivant cette ligne de travail, Chernova and Veloso (2009) proposent de demander un conseil à l'expert lorsque l'agent estime qu'il est possiblement dans un état critique et que certaines actions à sa disposition peuvent être mauvaises.

5.2.2 TAMER+RL

Le cadre TAMER (pour *Training an Agent Manually via Evaluative Reinforcement*) (Knox and Stone (2009)) a été introduit pour entraîner des agents apprenant en ligne uniquement à partir de conseils humains et sans récompense traditionnelle. Un agent TAMER évolue donc comme un agent RL dans un environnement donné par un ensemble d'états et navigue d'état en état en prenant des actions. L'interaction d'un agent TAMER avec son environnement et avec le conseiller humain est présentée en Figure 5.1. Comme un agent RL, l'agent TAMER reçoit une récompense $H(s, a)$ après avoir exécuté l'action a dans l'état s. Cependant, contrairement à l'agent RL, la récompense n'est pas conçue préalablement mais directement fournie par le conseiller. Ceci conduit à la principale différence entre l'apprentissage par renforcement et TAMER, dans le second cas l'agent n'essaie pas d'optimiser ses récompenses cumulées mais agit de manière gloutonne afin de maximiser sa récompense immédiate. L'idée sous-jacente à ce paradigme est que lorsqu'un humain juge l'action d'un agent, il prend en compte d'une manière ou d'une autre les conséquences futures de cette action. Ainsi, le but de l'agent TAMER dans un état s est tout d'abord de prédire correctement $H(s, a)$ pour toutes les actions $a \in \mathcal{A}$ et ensuite d'agir de manière gloutonne selon son estimation $\hat{H}(s, a)$. Les résultats des expériences menées dans Knox and Stone (2009), Warnell et al. (2017) montrent que pour la même tâche, un agent TAMER apprend plus vite qu'un agent RL traditionnel. Ceci peut être facilement expliqué par le fait que le signal donné à l'agent TAMER est beaucoup plus riche que le signal de récompense d'apprentissage de renforcement traditionnel. En effet, celui-ci est donné à chaque tour et pas uniquement lorsque la tâche est résolue.

Cependant, les signaux humains peuvent être mauvais car les humains, bien que bons à résoudre certaines tâches ne sont pas parfaits. Ils peuvent par exemple sous-estimer ou sur-estimer les conséquences de chaque action. Des extensions de l'algorithme TAMER ont donc été introduites afin de pouvoir combiner les récompenses TAMER avec celles de l'apprentissage par renforcement, menant au cadre TAMER+RL qui permet de récupérer le meilleur des deux mondes. Dans Knox and Stone (2010), plusieurs méthodes de mélange des signaux de récompense et des signaux TAMER

sont étudiées. De toutes les méthodes étudiées, il est montré expérimentalement que le plus efficace est d'agir directement sur la politique, sans biaiser l'approximation de la Q-fonction. Ainsi à chaque pas de temps t, la meilleure méthode est de prendre dans l'état s l'action a donnée par $a = \text{argmax}_{a'}[Q(s,a') + \alpha H(s,a')]$, où α est un paramètre de l'algorithme donnant plus ou moins de poids aux conseils. Faire décroître ce paramètre avec le temps permet donc de donner progressivement de moins en moins d'importance aux conseils humains.

Dans la prochaine section, nous étendons ce travail en fournissant un moyen d'intégrer les conseils TAMER biaisant la politique dans un contexte hors-ligne.

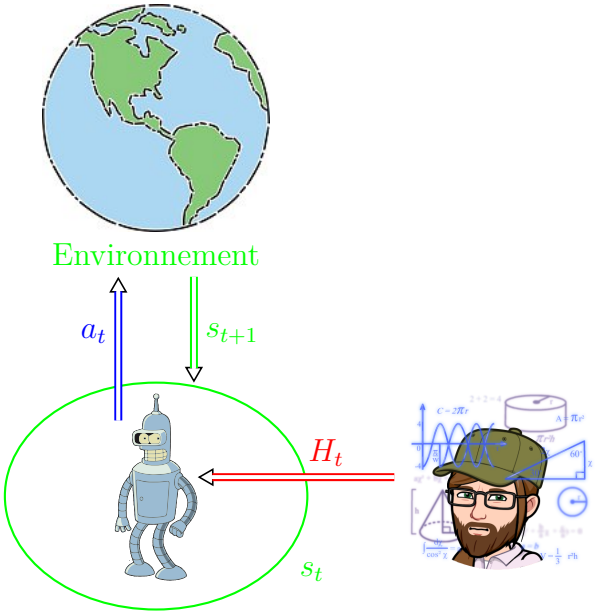


FIGURE 5.1 – TAMER

5.2.3 Algorithme TAMER-LSPI

Une procédure TAMER+RL hors-ligne peut être décomposée en quatre étapes comme présenté en Figure 5.2. Tout d'abord, un ensemble de données $\mathcal{D} = [(s_i, a_i, s'_i, r_i)]_{i=1}^N$ est collecté. Dans un second temps, cet ensemble est analysé et annoté par un expert humain. Ces annotations prennent la forme d'une fonction de récompense supplémentaire et permet de former un nouveau jeu de données $\mathcal{D}' = [(s_i, a_i, s'_i, r_i, H_i)]_{i=1}^N$, une récompense positive est donnée aux comportements efficaces et sûrs, tandis qu'une récompense négative est associée aux mauvais. Un algorithme TAMER+RL hors-ligne est ensuite appliqué à l'ensemble $\mathcal{D}'$ pour obtenir la meilleure politique possible à partir de cet ensemble. Il est important de souligner qu'un tel algorithme doit être efficace en termes de données par rapport aux annotations humaines. La supervision humaine est en effet très coûteuse et les annotations fournies par l'expert risquent d'être rares. Enfin, une fois qu'une stratégie a été générée par

l'algorithme Batch TAMER+RL, l'agent est relâché dans le monde réel où sa politique reste inchangée.

Fournir les conseils d'experts d'une manière hors-ligne est particulièrement intéressant d'un point de vue pratique. Il est en effet plus facile de déterminer a posteriori, *c'est-à-dire* quand toute la trajectoire est donnée, quelles étaient les actions qui ont conduit au succès (ou à l'échec) de cette trajectoire. Inversement, lorsque le retour de l'expert est donné en ligne, il n'est pas toujours évident de considérer les conséquences à long terme des actions de la machine. En outre, de manière hors-ligne, il est également plus facile de remarquer de bonnes tendances par rapport à ce qui a été observé et ainsi d'obtenir une fonction de récompense plus générale basée sur des heuristiques mettant l'accent sur ces tendances.

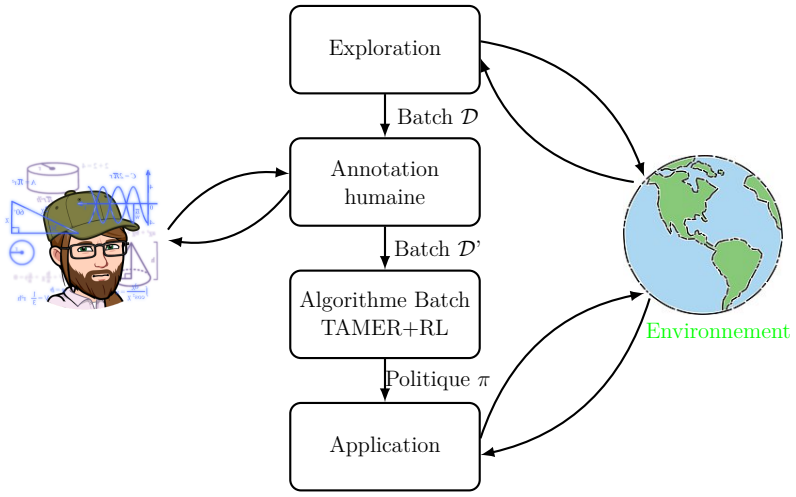


FIGURE 5.2 – Les quatre étapes d'une procédure hors-ligne TAMER+RL : 1. Récupération de données avec de l'exploration 2. Annotations des données par l'expert humain 3. Application d'un algorithme TAMER+RL hors-ligne 4. Application de la politique dans le monde réel.

Comme le montre Knox and Stone (2010), plus un algorithme affecte les Q-valeurs avec une récompense additionnelle (comme dans le cas du *Reward Shaping*), moins la prise en compte des conseils humains est efficace. À l'opposé, plus il affecte directement la politique, plus la politique obtenue est bonne. Nous avons donc conçu un algorithme TAMER+RL hors-ligne biaisant directement le mécanisme de sélection d'actions avec la récompense supplémentaire, tout en laissant les Q-valeurs inchangées. En tant qu'algorithme fonctionnant avec deux étapes indépendantes d'évaluation de politique et d'amélioration de la politique, LSPI est le candidat de choix comme base du nouvel algorithme. En effet, les modifications apportées à l'étape d'amélioration de la politique n'affectent pas la validité des valeurs apprises au cours de l'évaluation de la politique.

Pour faire face à la rareté des conseils fournis par les conseils humains et afin de pouvoir généraliser ces conseils à des états non annotés, H est également estimé

par une régression linéaire : $\hat{H}(s,a) = \langle w, \phi(s,a)\rangle$. Le vecteur de poids w est calculé en minimisant la perte : $\sum_{i=1}^{N}(H_i - \phi(s_i,a_i)w) + \lambda||w||_2$, où λ est un paramètre de régularisation ℓ_2, via la régression Ridge. De la même manière que l'apprentissage par renforcement hors-ligne fournit une amélioration majeure de la stabilité par rapport aux méthodes en ligne, la régression de la fonction H sur l'ensemble des données de manière synchrone conduit également à des estimations plus stables des valeurs H.

Comme indiqué en Section 2.2.3.2, pour apprendre une bonne politique à l'aide d'un algorithme basé sur la procédure d'itération de la politique, il est nécessaire de fournir pendant la phase d'amélioration des politiques une politique meilleure que la précédente. Dans notre cas, en suivant Knox and Stone (2010) et en supposant que les conseils humains sont *bons*, nous savons que, à chaque pas de temps t, en suivant l'action a, gloutonne par rapport à $Q^{\pi_{t-1}}(s,a') + \alpha H(s,a')$ conduit probablement à une amélioration de la politique globale, meilleure et plus sûre que celle obtenue en étant seulement glouton par rapport à la fonction de valeur apprise. $H(s,a)$ compense ainsi les erreurs d'approximation obtenues en estimant $Q^{\pi}(s,a)$.

Ces modifications mènent alors à l'Algorithme 10.

Algorithme 10 TAMER-LSPI

Entrées : Un ensemble de transitions $\mathcal{D} = \{s_i, a_i, s'_i, r_i, H_i\}_{i=1}^{N}$, une politique π_0, un seuil ϵ, $j = 0$, un paramètre α

Régresser la fonction H en minimisant la fonction de coût définie par $\sum_{i=1}^{N}(H_i - \phi(s_i,a_i)w) + \lambda||w||_2$ où λ est un paramètre de régularisation

Tant que $||Q^{\pi_k} - Q^{\pi_{k-1}}|| > \epsilon$ **faire**

$j = j + 1$

Évaluation de la politique

$(A - \hat{\gamma B})_0 = \underline{0}$, $\hat{b_0} = \underline{0}$

Pour $i = 1..N$ **faire**

$(A - \hat{\gamma B})_i = (A - \hat{\gamma B})_{i-1} + \phi(s_i,a_i)\phi^T(s_i,a_i)_\gamma \phi(s_i,a_i)\phi^T(s'_i,\pi(s'_i))$

$\hat{b_i} = b_{i-1} + \phi(s_i,a_i)r_i$

Fin pour

Résoudre $\frac{1}{N}(A - \hat{\gamma B})_N \theta = \frac{1}{N}\hat{b_n}$

Amélioration de la politique

$\forall s \in \mathcal{S},\ \pi_{n+1}(s) = \text{argmax}_{a\in\mathcal{A}}\, \hat{Q}^{\pi_n}(s,a) + \alpha\hat{H}(s,a)$ avec $\hat{H}(s,a) = w^T\Phi(s,a)$

Fin tant que

Retourner π_j

En diminuant le paramètre α à chaque passe de l'algorithme, l'impact du biais humain diminue et asymptotiquement, l'agent apprend la politique optimale.

Il est important de noter que la récompense supplémentaire n'est pas nécessairement une étiquette donnée par l'expert humain à chaque échantillon. Elle peut encoder toutes sortes de récompenses et ce sans prendre soin de la faire dériver d'une fonction potentielle d'états. Par exemple, elle peut correspondre à différents scores encodant des retours utilisateurs (El Asri et al. (2016b)), une estimation des Q-valeurs calculées sur un utilisateur simulé (Su et al. (2015)), ou encore une récompense basée sur des

signaux sociaux tels une expression ou un geste de l'interlocuteur (Ferreira and Lefevre (2015))... Dans la Section 5.3, un ensemble des règles encodées sous la forme d'une fonction de récompense est utilisé comme fonction TAMER.

5.3 Cadre expérimental

5.3.1 Tâche

Nous avons choisi de valider expérimentalement notre algorithme sur un simulateur de réservation de restaurant. Dans cette tâche, un système de dialogue doit aider son utilisateur humain à trouver un restaurant adapté à ses goûts personnels. Dans notre cadre, l'humain est modélisé par un simulateur d'utilisateur que nous décrivons en Section 5.3.3.

5.3.2 Le jeu de données DSTC2

Pour rendre l'expérience aussi réaliste que possible, nous avons entraîné tous nos modèles sur le jeu de données Dialogue State Tracking Challenge 2 (DSTC2) (Henderson et al. (2014)). Ce jeu de données contient 3000 dialogues humain/machine. Une caractéristique clé de cet ensemble de données est que tous les énoncés sont annotés avec des actes de dialogue. En outre, à chaque tour de dialogue, les objectifs de l'utilisateur sont explicitement donnés. Dans ce jeu de données, l'objectif du système de dialogue est d'aider l'utilisateur à trouver un restaurant correspondant à ses contraintes, puis de lui fournir toutes les informations qu'il pourrait requérir, comme l'adresse, le numéro de téléphone... La Figure 5.3 illustre le type de dialogues se trouvant dans le DSTC2.

Dans ce jeu de données, le Gestionnaire de Dialogue est remplacé par un Magicien d'Oz, ce qui permet de rassembler des données système expertes et de fournir un comportement d'utilisateur aussi conforme au dialogue humain/machine que possible.

5.3.3 Simulation d'utilisateur

Nous avons construit le simulateur d'utilisateur selon la méthodologie fournie par El Asri et al. (2016a). Dans ce cadre, l'utilisateur est modélisé par une LSTM prenant en entrée l'intégralité de l'historique des contextes de dialogue et génèrant l'acte de dialogue utilisateur le plus probable.

Pour construire le simulateur, au début de chaque dialogue, un objectif $G = (C, R)$ est tiré aléatoirement. Cet objectif contient les contraintes C et les requêtes R de l'utilisateur. Les contraintes sont les valeurs devant être données au système de dialogue, tandis que les requêtes sont les valeurs que l'utilisateur doit recevoir du système. L'utilisateur a au plus trois contraintes à donner au système et peut requérir trois informations.

Un contexte de dialogue c_t au pas de temps t est défini par les composantes suivantes : le dernier acte de dialogue a_t énoncé par le gestionnaire de dialogue, un vecteur

System	Bonjour, bienvenue dans le système de restaurant de Cambridge. Vous pouvez demander des restaurants par région, gamme de prix ou type de nourriture. Comment puis-je vous aider ?	`welcomemsg`
User	Restaurant suédois à prix modéré	`inform`[Swedish]
System	64/5000 Désolé, il n'y a pas de restaurant suédois dans la gamme de prix modérée	`canthelp`
User	Qu'auriez vous dans un type asiatique oriental ? ?	`inform`[Asian oriental]
System	Yippee noodle bar est un excellent restaurant servant de la cuisine orientale asiatique et il est dans la gamme de prix modérée	`offer`[Yippee noodle bar]
User	Pourrais-je avoir l'adresse et le numéro de téléphone ?	`request`[phone, address]
System	Bien sur , yippee noodle bar est au 40428 King Street City Centre	`offer`[address]
System	Le numéro de téléphone du yippee noodle bar est 01223 518111	`offer`[phone]
User	Merci, au revoir	`bye`

FIGURE 5.3 – Exemple de dialogue - La première colonne correspond au locuteur, la seconde est une transcription de l'énoncé et la troisième est l'acte de dialogue fourni en sortie du module de compréhension

d'incohérence i_t suivant les incohérences entre les informations fournies par la machine et l'utilisateur, un vecteur de contraintes c_t et un vecteur de requête r_t suivant les informations de l'objectif utilisateur données à la machine.

Le vecteur d'action machine a_t est un vecteur de taille n_{ma} où n_{ma} est le nombre d'actions disponibles de la machine. Ce vecteur a 1 pour chaque acte de dialogue produit par la machine dans le tour précédent et 0 sinon.

Le vecteur d'incohérence i_t est donné par deux vecteurs de taille n_c, où n_c est le nombre de contraintes possibles de l'utilisateur. Lorsque le système propose un restaurant à l'utilisateur, le premier vecteur vérifie que toutes les contraintes de l'utilisateur ont été respectées. Le deuxième vecteur suit les incohérences du gestionnaire de dialogue lorsqu'il mentionne une contrainte dans toutes les autres situations (*par exemple* dans une confirmation). Ces deux vecteurs ont des 1 pour chaque contrainte violée et des 0 partout ailleurs. Ces vecteurs sont réinitialisés après chaque tour.

Les vecteurs de contraintes et de requêtes c_t et r_t suivent quelles sont les valeurs de l'objectif utilisateur ayant déjà été données au système et correctement comprises par celui-ci. Au début de chaque dialogue, ces vecteurs ont des 0 pour chaque contrainte C et requête R dans l'objectif de l'utilisateur G et 0 partout ailleurs. Si une contrainte est mise à zéro dans le vecteur d'incohérence, elle est réinitialisée à un dans le vecteur de contrainte. Le vecteur de requête est réinitialisé après chaque proposition de la machine.

Ces vecteurs de contexte sont utilisés comme entrées pour alimenter une LSTM de 64 unités.

Nous ajoutons à cette LSTM une couche softmax afin de générer des probabilités sur les actions possibles de l'utilisateur. Ce réseau de neurones est ensuite entraîné sur l'ensemble de données DSTC2. Les actions de l'utilisateur sont finalement choisies en récupérant l'action la plus vraisemblable en fonction des probabilités fournies par le softmax.

Enfin, un ensemble de règles surmontant l'action choisie par le LSTM est rédigé manuellement afin d'assurer un comportement cohérent. Tout d'abord, puisque les conseils les plus importants que nous allons fournir au système concernent comment demander une confirmation, nous avons imposé à l'utilisateur de répéter sa dernière phrase lorsque le système le demande, et d'approuver ou de réfuter si le système demande une confirmation sur ce qu'il a compris. Deuxièmement, pour que le dialogue se termine, avec une probabilité $p = 0,02$ ou si l'utilisateur a donné/reçu toutes les informations dont il avait besoin, il termine le dialogue.

5.3.4 Espace d'états et modèle d'ASR

Afin d'appliquer des méthodes d'apprentissage par renforcement pour entraîner le système de dialogue, un espace d'état markovien doit être conçu. Dans les expériences suivantes, la méthode d'apprentissage par renforcement profond proposée par Bakker (2002) a été choisie pour en apprendre une. L'idée principale derrière cette approche est de considérer une LSTM similaire au simulateur d'utilisateur mais d'apprendre à générer les actions du système de dialogue présentes dans l'ensemble de données DSTC2. Puisque l'état de cellule de cette LSTM contient suffisamment d'informations pour prédire le futur, il respecte la propriété de Markov et peut être utilisé comme état du gestionnaire de dialogue.

Plus précisément, nous avons d'abord implémenté une LSTM produisant l'action la plus probable du système compte tenu de l'historique entier des contextes de dialogue. Ces contextes sont constitués de trois vecteurs, le vecteur d'action utilisateur a'_t, le vecteur de contraintes c'_t et le vecteur de requête r'_t. Le vecteur d'action utilisateur a'_t a n_{ua} composants, où n_{ua} est le nombre d'actions de dialogue système possibles. Ce vecteur a un pour chaque action de dialogue donnée par l'utilisateur au tour précédent et des zéros partout ailleurs.

Les vecteurs de contraintes et de requêtes c'_t et r'_t ont un pour chacune des contraintes données par l'utilisateur au système et des zéros partout ailleurs. Dans le cas d'une incompréhension de l'ASR, un emplacement aléatoire est mis à un dans le vecteur de contrainte.

De même que pour le simulateur d'utilisateur, une couche softmax est ajoutée au LSTM, les sorties de cette couche correspondant aux probabilités des différentes actions du système.

Expérimentalement, nous avons remarqué qu'une telle représentation était sujette au phénomène *internal covariate shift* (Ioffe and Szegedy (2015)), ce qui conduisait à

de mauvaises représentations de la fonction Q. Ce problème est surmonté en stabilisant la dynamique de l'état caché par des méthodes dites de *layer normalization* (Ba et al. (2016)).

Il est également supposé dans ce travail que le module ASR du système de dialogue n'est pas parfait et que des malentendus peuvent se produire. Lorsque l'utilisateur donne une de ses contraintes au système, avec une probabilité $p = 0.3$, le système comprend autre chose. Un score ASR est ensuite calculé en suivant la méthodologie de Khouzaimi et al. (2015) et présenté en Section 4.5.1.1. Un nombre aléatoire x est d'abord tiré selon une distribution normale centrée sur $+1$ dans le cas d'une compréhension correcte et centré sur -1 dans le cas d'un malentendu. Le score final est ensuite calculé en passant x à travers une fonction sigmoïde : $score = \frac{1}{1-e^x}$.

Nous avons supposé que tous les autres actes de l'utilisateur sont toujours correctement compris par le système. Dans ce cas, le score ASR est 1.

Puisque cette information sur le score ASR n'est pas présente dans le LSTM, nous avons construit l'espace d'état du système de dialogue en concaténant le vecteur d'état cellulaire c'_t de cette LSTM avec le score ASR et le pas de temps t.

5.3.5 Espace d'actions

L'espace d'action du système contient tous les différents actes de dialogue machine présents dans le corpus. Pour simplifier les choses, lorsque le système suggère un restaurant, nous avons supposé qu'il y a toujours un restaurant satisfaisant toutes les contraintes comprises par le système. Pour la même raison, si le système répond à une requête de l'utilisateur, on suppose qu'il peut toujours fournir une réponse à cette requête.

5.3.6 Fonction de récompense

Dans les expériences, nous avons utilisé un schéma de récompense traditionnel pour DSTC2. Si le dialogue se termine avec succès (*i.e.* un restaurant satisfaisant toutes les contraintes de l'utilisateur et si toutes ses demandes ont été répondues), une récompense de $+1$ est donnée au système. Cependant, si l'utilisateur (ou le système) raccroche avant de réaliser tous ses objectifs, aucune récompense n'est donnée au système.

Pour s'assurer que le système essayera de rendre le dialogue aussi court que possible, le facteur d'amortissement γ est fixé à 0,9.

5.3.7 Récompenses TAMER

Pour fournir de bons conseils humains, une politique simple et quasi-optimale est conçue manuellement. Dans l'état s, prendre l'action a prescrite par les règles conduit à une récompense TAMER $H(s, a)$ de $+1$, alors qu'une autre action n'entraîne aucune récompense supplémentaire.

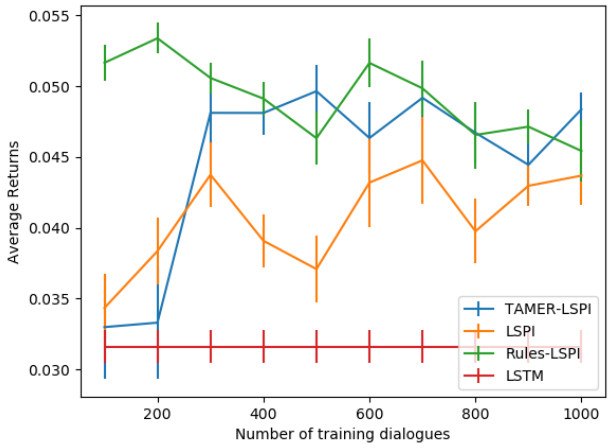


FIGURE 5.4 – Retour Moyen Amorti

Ces règles ne tiennent compte que du dernier acte de dialogue de l'utilisateur et du score ASR. Au début du dialogue, le système accueille l'utilisateur. Si le score ASR est inférieur à 0,3, il demande à l'utilisateur de répéter ce qui a été dit ; s'il est compris entre 0,3 et 0,7, il demande une confirmation ; et s'il est supérieur à 0,7, il demande à l'utilisateur de lui fournir une de ses contraintes ou une de ses requêtes. Si l'utilisateur demande à répéter, le système répète. Quand il/elle requiert quelque chose, le système l'en informe. Sinon, le système demande à l'utilisateur de lui fournir une de ses requêtes.

Finalement, afin d'éviter une fin précoce induite par le système, nous avons ajouté une récompense -1 TAMER si le système termine le dialogue.

5.4 Résultats

Nous avons choisi de comparer les performances de TAMER-LSPI avec LSPI comme algorithme pur d'apprentissage par renforcement hors-ligne, avec une politique obtenue en mélangeant règles et LSPI, et avec la politique générative fournie par la LSTM simulant le système. Dans toutes les expériences, le paramètre de régularisation de LSPI β est mis à 2 et le paramètre de régularisation TAMER λ est mis à 1. Dans tous les cas, un ensemble de 200 dialogues est d'abord collecté en suivant la stratégie basée sur les règles définies dans la section précédente. Chaque algorithme d'apprentissage est ensuite entraîné sur ces données. La politique obtenue est alors utilisée pour rassembler plus de données, 50 dialogues sont d'abord collectés avec cette politique et 50 dialogues en suivant une stratégie d'exploration ϵ-greedy, avec ϵ mis à 0.1. Ce processus d'apprentissage des politiques/collecte de données est refait sur 20 passes, jusqu'à ce que 2000 dialogues soient recueillis.

À chaque passe k de l'algorithme, le paramètre α de l'algorithme TAMER-LSPI est fixé à $1/k$. Le mélange règle/renforcement est fait en appliquant la politique apprise

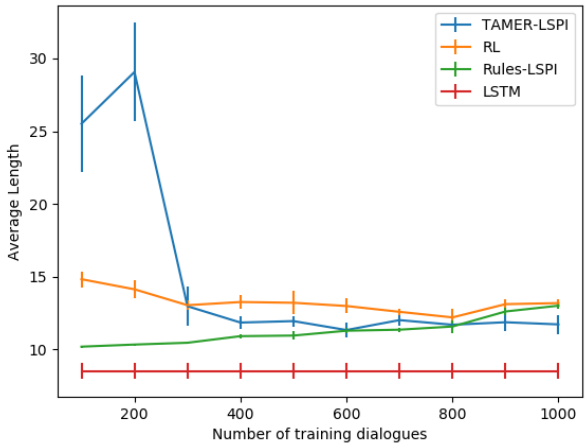


FIGURE 5.5 – Longueur des Dialogues

par renforcement et en choisissant l'action prescrite par les règles avec une probabilité $1 - 0.1 * (k - 1)$ si bien qu'au début, la politique est entièrement basée sur des règles tandis qu'à la fin, elle n'est basé sur l'apprentissage par renforcement. Afin de modéliser la non-optimalité des règles fournies dans un scénario de dialogue réel, nous y avons introduit une stochasticité. Ainsi, lorsque le système doit suivre les règles, avec une probabilité $p = 0.25$, il prend une action aléatoire non prescrite par les règles.

La Figure 5.4 montre les performances des quatre politiques à chaque passe de la procédure en termes de retours amortis, tandis que la Figure 5.5 montre les longueurs moyennes des dialogues. Les résultats sont calculés en faisant la moyenne des performances sur 20 passes où chaque algorithme est réentraîné depuis zéro.

Sur les deux figures, on constate qu'avec un petit nombre d'échantillons, les performances de la politique basée sur les règles et du renforcement surpassent les politiques fournies par la LSTM, LSPI et TAMER-LSPI. Cela s'explique facilement par le fait que la politique basée sur les règles est presque optimale et qu'aucun algorithme d'apprentissage ne peut apprendre une si bonne politique avec aussi peu d'échantillons.

D'un autre côté, alors que TAMER+RL est moins performant que la politique basée sur les règles aux premières passes, il fonctionne toujours mieux que l'apprentissage par renforcement pur. Ce résultat n'est pas surprenant car les informations supplémentaires fournies par les conseils permettent à TAMER-LSPI d'être meilleur que l'apprentissage par renforcement, manquant d'échantillons. Fait intéressant, il semble également fonctionner mieux que RL asymptotiquement. Ce phénomène pourrait s'expliquer par le fait que TAMER-LSPI a pu explorer des régions plus intéressantes de l'espace d'état au cours de la phase de collecte de données.

En outre, la Figure 5.5 montre qu'au début, la longueur moyenne du dialogue de TAMER-LSPI est plus longue que celle obtenue avec les autres algorithmes. Cette longueur de dialogue diminue drastiquement après environ 300 dialogues d'entraînement. Cela signifie que même si le système a appris au début de ne pas utiliser l'action ter-

miner le dialogue en raison des conseils, il apprend alors à aller au-delà de ces conseils et ne les prend en compte que lorsqu'ils sont bénéfiques, ce qui démontre la robustesse contre les mauvais conseils humains.

5.5 Conclusion

Dans ce chapitre, nous avons introduit une nouvelle méthode, basée sur le cadre TAMER + RL, pour incorporer des conseils humains durant la phase de formation des systèmes de dialogue statistique afin de fournir un comportement sûr. Les résultats expérimentaux montrent que l'utilisation de tels signaux peut affecter les performances des algorithmes d'apprentissage par renforcement à tous les stades de la phase d'apprentissage, au début où le manque d'échantillons d'entraînement conduit souvent à des politiques de RL mauvaises et dangereuses, mais asymptotiquement. Les résultats montrent également que TAMER + RL est robuste contre les mauvais préjugés humains. Notre méthode présente finalement l'avantage d'être complètement statistique. Il est donc compatible avec des modèles d'Apprentissage Profond plus complexes qui devraient être capables de gérer des situations réelles où aucune politique artisanale efficace ne peut être conçue et où les conseils peuvent être parcimonieux (*i.e.* seulement quelques règles accompagnées d'échantillons étiquetés).

Encore une fois, la représentation de l'humain sous la forme d'un environnement stationnaire n'était pas suffisante pour aborder le cadre considéré. Nous allons au chapitre suivant voir une troisième manière de modéliser l'humain par le biais d'un autre problème : le dialogue humain/humain assisté par ordinateur.

Chapitre 6

Le dialogue humain/humain assisté par ordinateur

Au cours des chapitres précédents, nous avons vu deux scénarios dans lesquels il n'était pas suffisant de modéliser l'être humain sous la forme d'une distribution de probabilité stationnaire. Tout d'abord, l'être humain a été vu comme un agent apprenant, capable de s'adapter aux évolutions de son système de dialogue. Dans un second temps, nous avons vu de quelle manière ses préférences, présentées sous la forme de conseils, pouvaient être prises en compte durant la phase d'optimisation du système de dialogue, permettant un apprentissage plus rapide et plus sûr. Dans ce chapitre, nous abordons une troisième situation dans laquelle le modèle d'environnement stationnaire est a priori adapté mais dans laquelle, compte tenu des spécificités de l'humain, les traditionnelles fonctions de récompense évaluant la réussite d'une tâche ne conviennent pas totalement. Plus précisément, nous considérons une situation dans laquelle deux humains discutent et un système de dialogue écoute la conversation pour fournir des informations contextuelles aux interlocuteurs.

6.1 Motivations

Les dialogues orientés tâche ne sont pas une spécificité du dialogue humain/machine. En effet, négocier pour prendre un rendez-vous, interagir avec un télé-opérateur dans l'objectif de résoudre un problème de ligne téléphonique, ou prendre des décisions lors d'une réunion sont des dialogues humain/humain orientés tâche. Dans certains cas, un interlocuteur a besoin d'informations externes pour effectuer la tâche avec succès. Dans le cas du dépannage de la ligne téléphonique par exemple, le télé-opérateur peut avoir besoin d'informations techniques sur le problème du client pour fournir un support efficace. Le temps pris pour rechercher cette information peut détériorer la fluidité du dialogue, menant à un mauvais retour utilisateur.

Dans ce chapitre, nous nous intéressons au problème du dialogue humain/humain assisté par ordinateur. Ici, la tâche de fournir des informations contextuelles aux interlocuteurs est automatisée par l'intermédiaire d'un *assistant de conversation* (Conversational Assistant en anglais, CA)). Nous cherchons ici à donner un cadre général au problème. Dans ce chapitre, nous ne faisons pas d'hypothèses sur la manière dont l'information est transmise par le CA. Celui-ci peut en effet agir sur la conversation de différentes manières comme par exemple la prise de parole, l'ajout de fenêtres pop-ups sur un écran *etc.*

Pour aborder ce problème, une analyse du dialogue humain/humain est nécessaire. Si celui-ci a été étudié depuis longtemps du point de vue des sciences cognitives/sociales (Austin (1975), Clark and Schaefer (1989)), son traitement computationnel est relativement récent. Des travaux du milieu des années 1990 comme Gorin et al. (1997), Waibel et al. (1998) présentent des systèmes capables d'écouter passivement des conversations humain/humain et de les traiter en arrière-plan (le premier proposant d'acheminer un client vers un interlocuteur plus compétent sur son problème et le dernier résumant automatiquement des réunions). Depuis, la recherche sur le dialogue humain/humain s'est principalement concentrée sur des tâches de compréhension du langage parlé telles l'étiquetage d'actes de dialogue (Stolcke et al. (2000)), la détection de sujet (Hsueh et al. (2006)), le suivi de l'état du dialogue (Kim et al. (2016)), *etc.*

Dans ce chapitre, nous nous intéressons à un aspect différent du problème en abordant les difficultés dues la nature séquentielle de la tâche. Ainsi, nous prenons en compte le fait que les énoncés du CA auront un impact direct sur le futur de la conversation. Comme dans le cas du dialogue humain/machine, dans le dialogue humain/humain assisté par ordinateur, la gestion de dialogue peut être vue comme un problème de décision séquentielle, où les décisions sont prises sur quel acte de dialogue doit être choisi selon le contexte dialogique actuel.

La problématique de la conception des algorithmes permettant d'assister les humains dans l'objectif d'améliorer leur comportement au cours du temps a été abordée dans de nombreux contextes. Ainsi par exemple, l'enseignement machine (ou *Machine Teaching* (Simard et al. (2017)) en anglais ou *Curriculum Learning* (Bengio et al. (2009))) abordent le problème de la sélection d'exemples à présenter à un être humain pour accélérer son apprentissage d'une tâche comme par exemple la maîtrise d'une langue (Daubigney et al. (2013), Settles and Meeder (2016)). Ces travaux supposent cependant que, pour la tâche considérée, l'être humain est un agent apprenant et que lui présenter des exemples lui permettra d'optimiser son comportement.

Cette hypothèse n'est cependant pas réaliste dans notre cas. Les humains n'ont en effet a priori pas besoin d'apprendre à mieux converser, ils ont simplement besoin d'un contexte de conversation plus riche. La manière de s'insérer dans la conversation doit cependant être abordée dans ce cas. En effet, il est raisonnable de penser que dans une conversation entre deux humains, ceux-ci ne souhaitent pas que l'influence de la machine soit trop important. En effet, si la machine interrompt trop souvent les êtres humains par exemple, la fluidité de la conversation serait en péril et il se pourrait que les humains décident tout simplement de se passer de la machine. Ce problème a à notre connaissance été très peu abordé. Meguro et al. (2013) aborde cependant un problème relativement proche nommé dialogue humain/machine basé sur l'écoute. Ici, à l'instar du système de psychanalyse ELIZA (Weizenbaum (1966)), on cherche à laisser parler l'humain le plus longtemps possible, en ayant aussi peu d'influence que possible sur la suite de la conversation. Dans ce travail, une récompense est donnée à la machine si le dialogue réussit et une récompense additionnelle est donnée si celle-ci choisit l'action maximisant la probabilité que la réponse de l'humain ait été la réponse fournie si son interlocuteur était également humain.

Dans ce chapitre, nous proposons également de modéliser la tâche de dialogue humain/humain assisté par ordinateur comme un problème d'apprentissage par renforcement, dont une composante de la fonction de récompense est associée à la réussite du dialogue. Une seconde composante est ajoutée à celle-ci rendant compte de l'influence de l'action de la machine sur le dialogue. De cette manière, un juste milieu entre efficacité et intrusivité est appris. Comme nous le verrons, notre méthode permet de modéliser le problème sous la forme d'un processus de décision de Markov linéairement solvable (Todorov (2006)), dont il existe des moyens efficaces de résolution.

Ce chapitre est organisé de la façon suivante. Tout d'abord, un modèle général de l'interaction humain/humain est établi. Nous abordons ensuite la problématique de la perturbation de cette interaction. Finalement, nous confrontons notre approche à l'expérience.

6.2 Un modèle général de l'interaction humain/humain

6.2.1 Séquences temporelles

Du point de vue d'un observateur extérieur (comme par exemple celui d'un assistant de conversation), un dialogue peut être vu, ainsi que montré par la Figure 6.1, comme une séquence d'observations o_t, appartenant à un certain ensemble $\mathcal{O}$. Cet ensemble peut par exemple être l'ensemble des énoncés possibles dans le cas d'un observateur humain ou celui des actes de dialogue ajoutés à des scores de confiance dans le cas d'un gestionnaire de dialogue. Il peut être supposé que chaque dialogue $d = \{o_0, o_1, ..., o_t, ...\}$ est tiré selon une certaine distribution de probabilité :

$$p(o_0, o_1, ...) = p(o_0) \prod_{t>0} p(o_t|h_t), \tag{6.1}$$

où $h_t = \{o_0, ..., o_{t-1}\}$ est appelé l'*historique* au pas de temps t.

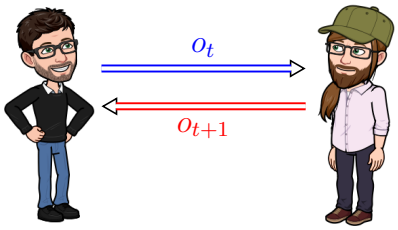


FIGURE 6.1 – Dialogue humain/humain

6.2.2 Chaînes de Markov

De la même manière que dans le cas des MDP développé en Section 3.2.1, l'espace des historiques est en pratique très grand, si bien qu'échantillonner directement à partir de cette distribution est généralement impossible en un temps raisonnable. Cependant, il est possible de compresser l'information pertinente contenue dans ces historiques sous la forme d'un état appartenant à un ensemble appelé espace d'états $\mathcal{S}$. Ces états sont construits de manière à vérifier l'hypothèse de Markov (le futur dépend seulement du présent et est indépendant du passé). En pratique, il est toujours possible de représenter une séquence temporelle sous la forme d'une chaîne de Markov en définissant les états comme les historiques $s_t = h_t$. De la même manière que dans le cas du dialogue humain/machine, concevoir des représentations d'états plus sophistiquées, étant à la fois compactes et respectant la propriété de Markov, demeure toujours un important domaine de recherche (Kim et al. (2016)).

Ainsi, à partir d'une représentation d'états donnée, un dialogue peut être vu comme une séquence d'états $d = \{s_0, s_1, ..., s_t, ...\}$ tirés selon une certaine distribution de probabilité appelée fonction de transition :

$$p(s_0, s_1, ...) = p(s_0) \prod_{t>0} p(s_t|s_{t-1}). \tag{6.2}$$

Un tel processus est appelé chaîne de Markov.

Dans ce chapitre, nous supposons que les états sont données. Cependant, la fonction de transition entre les états est inconnue.

6.2.3 Processus de Markov valué

Bien que les chaînes de Markov sont un outil puissant pour modéliser le dialogue, elles ne permettent pas son évaluation. De la même manière que dans le cas des MDP, on peut définir une *fonction de récompense*, associant l'espace d'états à l'ensemble de nombres réels. Une fonction de récompense associée à une chaîne de Markov définit un Processus de Markov valué (en anglais, *Markov Reward Process*). Formellement, une fonction de récompense associe un nombre réel à chaque état ($r : \mathcal{S} \to \mathbb{R}$) [1].

En supposant que chaque trajectoire termine presque sûrement, il est possible de définir une valeur à chaque état s comme étant l'espérance de la somme cumulée des récompenses obtenues en suivant le processus à partir de cet état. Formellement :

$$\forall s \in \mathcal{S},\ V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} r(s_t) \middle| s_0 = s\right]. \tag{6.3}$$

En d'autres termes, la fonction de récompense définit la qualité de chaque énoncé. Tandis que la fonction de valeur est une mesure quantifiant à quel point il est intéressant d'être dans un certain état.

1. La fonction de récompense ici définie est différente la fonction de récompense des MDP, qui est une fonction de l'espace des couple états-actions dans l'ensemble des nombres réels. Par souci de simplicité, nous confondons les notations dans ce chapitre.

Dans ce cas, la fonction de récompense définie correspond aux préférences d'un être humain. Comme nous l'avons vu au Chapitre 4.2, celle-ci peut différer selon que l'on considère l'un ou l'autre des interlocuteurs. Dans des cas de dialogues très simples, celle-ci peut être définie préalablement par des experts humains par exemple avec des outils comme PARADISE (Walker et al. (1997)), mais dans le cas d'un dialogue plus complexe, elle doit être inférée à partir de données, avec l'aide de l'apprentissage par renforcement inverse par exemple (Ng et al. (2000)) ou alors par régression de scores (El Asri et al. (2016b)).

Dans ce chapitre, nous considérons que le problèmes de l'apprentissage d'une fonction de récompense encodant les préférences de l'utilisateur du CA a été résolu et qu'une telle fonction est donnée.

6.3 Contrôler les chaînes de Markov

Notre objectif est de proposer un cadre afin d'améliorer le résultat d'un dialogue humain/humain par l'introduction d'un assistant de conversation. Nous voyons dans cette section comment les processus de Markov à récompense introduits dans la section précédente peuvent nous aider à résoudre ce problème.

6.3.1 Perte de Kullback-Leibler

Notre tâche peut être vue comme une tâche de décision dans l'incertain. En effet, comme nous l'avons vu, la conversation humain/humain est un processus stochastique et agir sur celle-ci modifie son état (ne pas agir peut être vu comme prendre l'action nulle). Nous pouvons donc modéliser notre tâche d'apprentissage d'une stratégie d'intervention comme un problème d'optimisation d'un MDP. Mais pour cela, une fonction de récompense doit être définie.

Nous supposons tout d'abord que les états de dialogue sont naturellement tirés à partir d'une distribution de probabilité P_0 lors de la conversation humain/humain. Si le système prend l'action a dans un état s_t, l'état suivant ne sera plus échantillonné selon P_0 mais selon une autre distribution P_a. Nous supposons également que les humains sont des quasi-experts en interaction et n'ont besoin que de petits changements d'état pour être optimaux, de petits ajouts d'informations suffisent pour que la conversation se passe au mieux. Ainsi, les actions choisies par le CA doivent se baser sur la capacité naturelle d'un humain à interagir, l'amélioration apportée portera sur une augmentation de ses connaissances (et donc du contexte de la conversation) mais ne modifiera pas trop radicalement ce contexte. Formellement, cela peut se traduire par le fait que l'on aimerait avoir P_a proche de P_0.

Cette proximité entre P_0 et P_a peut se mesurer grâce à la divergence Kullback-Leibler (KL), une mesure de similarité couramment utilisée pour mesurer la dissemblance entre deux distributions de probabilité. Formellement, si p et q sont deux distributions de probabilités sur un ensemble discret $\mathcal{S}$, la divergence KL entre p et q est définie par $\mathrm{KL}(p||q) = \sum_{s\in\mathcal{S}} p(s) \log \frac{p(s)}{q(s)}$. De manière moins formelle, la divergence KL

mesure à quel point un échantillon tiré de la distribution p aurait pu être tiré par la distribution q.

Cette mesure de similarité va nous aider à construire une fonction de récompense nous permettant de trouver un juste milieu entre trop modifier une interaction et améliorer de manière significative son résultat. Cet équilibre est donné en construisant cette fonction de récompense comme la somme de la mesure de la qualité actuelle de l'état avec la mesure de la perturbation induite par l'intervention de la machine. La fonction de récompense est donc définie de la façon suivante, prendre l'action a dans un état s induit une récompense :

$$r(s,a) = r(s) - KL(P_a(s,.)||P_0(s,.)).$$

6.3.2 Processus de Décision de Markov Linéairement Solvables

Dans cette section, nous décrivons un cadre efficace permettant de résoudre le problème d'optimisation défini précédemment : les Processus de Décision de Markov Linéairement Solvables (*Linearly Solvable Markov Decision Process* en anglais, LMDP, (Todorov (2006))). Les LMDP sont une classe spéciale de MDP traitant de la perturbation d'un processus de récompense de Markov sous un coût KL. Ces processus présentent l'intérêt que, en faisant une nouvelle hypothèse que nous relâcherons plus tard, ils rendent l'équation de Bellman linéaire. Ils sont donc plus facilement résolubles que les MDP classiques. En raison de cette propriété, leur efficacité a été démontrée sur une grande variété de problèmes tels la robotique (Matsubara et al. (2014), Ariki et al. (2016)), la production participative (Abbasi-Yadkori et al. (2015)), ou l'apprentissage en ligne (Guan et al. (2014), Neu and Gómez (2017)).

Les LMDP fonctionnent dans un cadre similaire au nôtre. Ils font cependant l'hypothèse que l'espace d'action est le simplex défini sur l'ensemble de la probabilité de transition $\mathcal{P}$, c'est-à-dire que les actions consistent à modifier directement les probabilités de transition et que toutes ces probabilités sont atteignables.

Dans ce cadre, en écrivant V_c^* comme la fonction de valeur optimale, l'équation d'optimalité de Bellman peut être réécrite comme suit :

$$\begin{aligned} V_c^*(s) &= \max_{P\in\mathcal{P}} \Big\{ r(s,P) + \sum_{s'\in\mathcal{S}} P(s,s')V_c^*(s') \Big\}, \\ &= \max_{P\in\mathcal{P}} \Big\{ r(s) - \mathrm{KL}(P(s,.)||P_0(s,.)) + \sum_{s'\in\mathcal{S}} P(s,s')V_c^*(s') \Big\}. \end{aligned}$$

Dans chaque état, la transition de fonction optimale est alors donnée par :

$$\begin{aligned} \pi^*(s) &= \operatorname*{argmax}_{P\in\mathcal{P}} \left\{ -\sum_{s'\in\mathcal{S}} P(s,s') \log \frac{P(s,s')}{P_0(s,s')} + \sum_{s'\in\mathcal{S}} P(s,s')V_c^*(s') \right\}, \\ &= \operatorname*{argmin}_{P\in\mathcal{P}} \left\{ \sum_{s'\in\mathcal{S}} P(s,s') \log \frac{P(s,s')}{P_0(s,s')e^{V_c^*(s')}} \right\}, \end{aligned}$$

$$= \underset{P \in \mathcal{P}}{\operatorname{argmin}} \, \mathrm{KL}\Big(P(s,.) \Big|\Big| \frac{P_0(s,.)e^{V_c^*(.)}}{Z(s)} \Big).$$

où $Z(s) = \sum_{s' \in \mathcal{S}} P_0(s,s') e^{V_c^*(s')}$ est une constante de normalisation.

En utilisant la propriété de la divergence KL statuant que celle-ci est minimale si et seulement si les deux distributions de probabilité sont identiques, la fonction de transition optimale est donnée par :

$$\pi^*(s) = P^*(s,.) = \frac{P_0(s,.)e^{V_c^*(.)}}{Z(s)} \tag{6.4}$$

L'équation de Bellman peut alors être réécrite par :

$$\begin{aligned}
V_c^*(s) =& r(s,P^*) + \sum_{s' \in \mathcal{S}} P^*(s,s') V_c^*(s') \\
=& r(s) - \sum_{s' \in \mathcal{S}} P^*(s,s') \log \frac{P^*(s,s')}{P_0(s,s')} + \sum_{s' \in \mathcal{S}} P^*(s,s') V_c^*(s') \\
=& r(s) + \sum_{s' \in \mathcal{S}} P^*(s,s') \log \Big(\frac{P_0(s,s')}{P^*(s,s')} e^{V_c^*(s')} \Big) \\
=& r(s) + \sum_{s' \in \mathcal{S}} P^*(s,s') \log Z(s) \\
=& r(s) + \log Z(s).
\end{aligned}$$

En passant cette équation à l'exponentielle et en la réécrivant sous forme vectorielle, l'équation de Bellman devient linéaire :

$$\forall s \in \mathcal{S}, \; e^{V_c^*(s)} = e^{r(s)} Z(s) = e^{r(s)} P_0(s,.) e^{V_c^*}.$$

6.3.3 Résoudre les LMDPs avec la Descente de Gradient Stochastique

Résoudre l'équation précédente avec des opérations d'algèbres linéaires est coûteux lorsque l'espace d'état est grand. Abbasi-Yadkori et al. (2015) introduisent cependant une méthode originale pour résoudre ce problème dans laquelle, au lieu de résoudre directement l'équation de Bellman passée à l'exponentielle $e^{V_c^*} = e^{LV_c^*}$, la quantité suivante est minimisée $|e^{V_c^*} - e^{LV_c^*}|$.

Pour cela, la fonction de valeur passée à l'exponentielle est linéairement paramétrée : $\forall s \in \mathcal{S}, \; e^{V_c^*(s)} = \langle \theta, \phi(s) \rangle$ où, $\phi(s)$ est un vecteur de caractéristiques, θ un vecteur dont les composantes auront à être apprises et $\langle \theta, \phi \rangle$ le produit scalaire entre les vecteurs θ etϕ.

La valeur $e^{V_c^*(s)} - e^{r(s)} P_0(s,.) e^{V_c^*}$ peut alors être réécrite comme $\langle \theta, \phi(s) \rangle - e^r P_0(s,.) \langle \theta, \phi \rangle$ qui est linéaire et donc convexe.

Étant donnée une constante de régularisation positive H, $\mathcal{T}$ un ensemble de trajectoires et ν une distribution de probabilité sur ces trajectoires, il peut être montré que

l'optimum θ^* de la fonction de coût suivante :

$$c(\theta) = -\log(\langle\theta, \phi(s_0)) + H \sum_{t\in\mathcal{T}} \nu(T) \sum_{s\in T} \left| \langle\theta, \phi(s)\rangle - e^{r(s)} P_0(s, .)\langle\theta, \phi\rangle \right|$$

induit une fonction de valeur quasi-optimale, menant alors à une politique également quasi-optimale (Abbasi-Yadkori et al. (2015)). Étant donné que $c(\theta)$ est convexe, il est possible de l'optimiser par descente de gradient stochastique (Stochastic Gradient Descent en anglais, SGD), où une estimation non biaisée de la fonction de coût est donnée par :

$$\begin{aligned} r(\theta) = &-\frac{1}{\langle\theta, \phi(s_0)\rangle}\phi(s_0) \\ &+ H \sum_{s\in T} \left[\text{sign}\Big(\langle\theta, \phi(s)\rangle - e^{r(s)} P_0(s, .)\langle\theta, \phi\rangle \Big) \times \Big(\phi(s) - e^{r(s)} P_0(s, .)\phi \Big) \right]. \end{aligned}$$

6.3.4 Cas des actions discrètes

Dans le cas que nous considérons, l'assistant de conversation ne peut malheureusement accéder à l'ensemble des états de dialogue possible, seul un petit nombre est accessible par le biais d'un ensemble fini d'actions discrètes.

Dans ce cas, en suivant les équations précédentes avec un ensemble d'actions discrètes $\mathcal{A}$, l'action optimale est donnée par :

$$a^*(s) = \underset{a\in\mathcal{A}}{\text{argmin}}\, \text{KL}\Big(P_a(s, .) \Big|\Big| \frac{P_0(s, .)e^{V(.)}}{Z(s)} \Big).$$

Malheureusement, cette quantité n'est généralement pas calculable car la fonction de valeur V considérée est a priori inconnue. Todorov (2009) a cependant montré empiriquement que si l'on considère la politique prenant l'action gloutonne par rapport à la fonction de valeur V_c^* optimale dans le cas du MDP continu, la fonction de valeur V_d associée à cette politique ne sera pas loin de la fonction de valeur optimale V^*.

Dans la prochaine section, nous montrons l'efficacité et la pertinence de cette approche sur un problème de dialogue en simulation.

6.4 Cadre expérimental

La tâche dialogique considérée dans ce chapitre est une variante du jeu de dialogue de négociation introduit dans Barlier et al. (2015) et affiné dans Laroche and Genevay (2016) où deux humains interagissent afin de fixer un rendez-vous. Nous modifions ce problème en ajoutant à cette conversation un Assistant de Conversation. Nous considérons que celui-ci appartient à l'un des humains et qu'il est capable d'intervenir durant le dialogue pour optimiser le résultat général de celui-ci, défini comme le score obtenu par son propriétaire.

Humain 1	Souhaiterais tu un rendez-vous au créneau 3 ?
Humain 2	Non. Souhaiterais tu un rendez-vous au créneau 2 ?
CA	Tu devrais refuser.
Humain 1	Non. Souhaiterais tu un rendez-vous au créneau 5 ?
Humain 2	Oui, parfait.

FIGURE 6.2 – Exemple de dialogue correspondant au jeu de négociation avec assistant de conversation

6.4.1 Le jeu de dialogue de négociations

Présentons tout d'abord les règles de ce jeu de négociation. Nous considérons deux humains, indexés par $i = 1, 2$, devant interagir afin de convenir d'un rendez-vous. Pour cela, ils doivent s'entendre sur un créneau horaire τ parmi les N créneaux disponibles. Réserver le créneau horaire τ^i coûte $c^i_{\tau^i}$ à l'humain i tandis que réussir à s'accorder sur un rendez-vous $\tau^i = \tau^j$ lui fait recevoir une récompense R^i, menant ainsi à une récompense totale $r^i_\tau = \mathbb{1}_{\tau^i=\tau^j} R^i - c^i_\tau$.

Ce jeu de dialogue de négociation est un jeu au tour par tour dont le premier joueur est choisi au hasard. A chaque tour, 3 actions sont à sa disposition. Il peut accepter la dernière proposition en choisissant l'action `Accepter`. Cette action met fin au dialogue et chaque joueur reçoit sa récompense correspondant à ce créneau. Dans le cas contraire, l'utilisateur peut refuser la dernière proposition et suggérer un nouvel emplacement en utilisant l'action `RefProp`. Enfin, il peut prendre l'action `FinDial`, en terminant le dialogue sans donner de récompense à chacun. La Figure 6.2 donne un exemple de dialogue que l'on peut obtenir avec ce jeu.

Pour éviter les effets de bord, deux états sont artificiellement ajoutés, un état initial dans lequel tous les dialogues commencent et un état final dans lequel tous se terminent.

6.4.2 Stratégies

De la même manière que présenté dans Laroche and Genevay (2016), nous supposons que les stratégies des humains sont fixes et quasi-optimales (mais pas nécessairement optimales). Elles peuvent donc être implémentées à partir de règles.

Ces décisions sont prises en fonction de l'ensemble des créneaux disponibles associés à une récompense positive pour chaque agent. Lorsque cet ensemble est vide, nous considérons que l'agent termine le dialogue. Dans le cas contraire, il compare la récompense induite par la dernière offre avec celle associée au meilleur emplacement disponible. Si cette récompense est plus petite, l'humain refuse la proposition et suggère ce meilleur créneau disponible. Sinon, il accepte l'offre. Afin d'ajouter une certaine variabilité au simulateur, rendant compte de la variété des comportements humains, ces politiques sont randomisées. Si la dernière action était `RefProp`, avec la probabilité ϵ_1, l'utilisateur prend une action aléatoire entre `Accept`, `Enddial` et `Refprop`, dans ce cas, l'humain suggère le meilleur créneau à sa disposition.

6.4.3 Représentation d'états

Une représentation de l'espace d'états de la conversation respectant la propriété de Markov doit être conçue. Cette représentation correspond au point de vue de l'Assistant de Conversation. Tout d'abord, nous considérons que celui-ci a accès au calendrier de son propriétaire, induisant la fonction de récompense du dialogue.

L'Assistant de Conversation peut également suivre le dernier énoncé prononcé. Pour les mêmes raisons que celles exposées aux Chapitres 4 et 5, nous introduisons du bruit sur les entrées du CA. Ainsi, le type de chaque action est toujours correctement compris, mais lorsqu'une offre est faite, il peut y avoir un malentendu sur le créneau horaire entendu. Soit SER le taux d'erreur de la phrase du système. Avec la probabilité SER, le système comprend un mauvais créneau, alors qu'avec la probabilité $1-SER$ l'énoncé est reçu sans interruption. Un score de confiance est alors généré comme dans les chapitres précédents en posant : $score_{reco} = \frac{1}{1+e^{-X}}$ où $X \sim \mathcal{N}(c, 1)$, avec $c = 1$ si le joueur a compris la bonne option et $c = -1$ sinon.

Afin d'obtenir une représentation d'état entièrement markovienne, le nombre d'énoncés prononcés pendant le dialogue est également suivi.

6.4.4 Espace d'actions

Dans ce jeu, nous considérons que l'Assistant de Conversation dispose de quatre actions : `DevraitAccepter`, `DevraitTerminerDialogue`, `DevraitRefuser` ou ne rien faire. Ces actions sont prises juste avant le tour de son propriétaire et représentent les suggestions du système. Pour modéliser le fait que l'utilisateur humain n'est pas obligé de suivre ces suggestions, nous considérons que si le système suggère une action, son utilisateur prend cette action avec une probabilité ϵ_2 tandis que dans le cas contraire, il choisit une autre action de manière aléatoire.

6.5 Résultats

Dans nos expériences, nous considérons des emplois du temps de chaque créneau. Les coûts pour chaque créneau sont échantillonnés de manière aléatoire entre 0 et 4, tandis que la récompense obtenue lorsqu'un rendez-vous est finalement fixé est de 3, ce qui conduit à environ 40000 états.

Le paramètre de stochasticité du comportement humain est fixé à $\epsilon_1 = 0.3$ tandis que le paramètre modélisant l'influence des suggestions de l'assistant de conversation sur son utilisateur est fixé à $\epsilon_2 = 0.9$. Pour traiter des espaces d'états discrets, nous divisons le score de confiance en 6 catégories autour des points de coupure 0,05, 0,15, 0,5, 0,85, 0,95 et 1.

Les composantes du vecteur de caractéristiques sont
— une valeur binaire indiquant si l'on se trouve dans l'état initial ou l'état final
— les récompenses passées à l'exponentielle associées à chaque créneau
— le nombre de tours passés

— la récompense obtenue passée à l'exponentielle si la dernière action était `Accepter`
— l'exponentielle de la différence de récompenses entre l'emplacement suggéré et les autres emplacements
— les valeurs passées à l'exponentielle des créneaux
— la valeur d'accepter l'emplacement suggéré si la dernière action était `RefProp` et si c'est le tour de l'utilisateur principal
— la récompense exponentielle de chaque emplacement disponible et la valeur de la dernière action suggérée
— une caractéristique de biais

La Descente de Gradient Stochastique a été appliquée sur 20000 itérations avec un taux d'apprentissage constant $\eta = 1$ pendant les 5000 premières itérations et $\eta = 0.1$ pour les autres. À chaque étape, ce taux d'apprentissage est divisé par la norme du gradient. Le paramètre de régularisation est défini par $h = 5$. Pour accélérer la convergence, nous avons utilisé l'astuce du mini-batch en faisant la moyenne à chaque itération d'un gradient calculé sur 5 échantillons tirés aléatoirement. La matrice de transition P_0 et les divergences KL entre les distributions sont estimées par la méthode de Monte-Carlo sur 20000 trajectoires pour chaque état.

Nous choisissons de représenter en Figure 6.3 la récompense moyenne (calculée sur 1000 dialogues) par l'humain avec l'aide de l'assistant de conversation en fonction du taux d'erreur phrase. La ligne pointillée représentant la récompense moyenne obtenue sans CA. En Figure 6.4, nous représentons le nombre moyen d'actions prises par dialogue en fonction du taux d'erreur phrase.

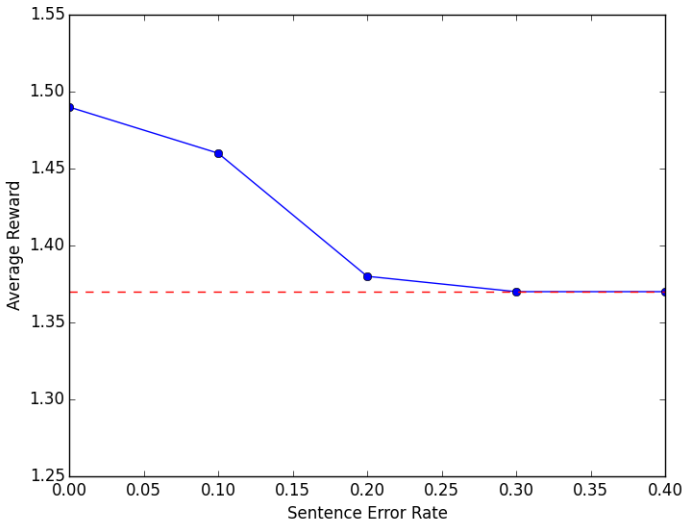


FIGURE 6.3 – Récompense moyenne obtenue fonction du taux d'erreur phrase

Nous observons en Figure 6.3 que la récompense moyenne diminue progressivement avec un SER croissant, montrant qu'en présence de bruit, la précision du système diminue. Nous observons cependant que dans tous les cas, la récompense obtenue n'est

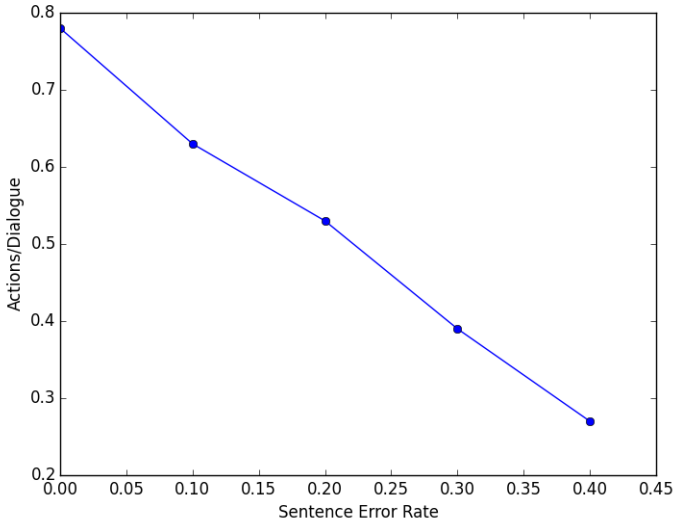


FIGURE 6.4 – Nombre moyen d'interventions du CA par dialogue fonction du taux d'erreur p̂hrase

jamais pire que la récompense moyenne sans aide. Sur la Figure 6.4, nous voyons qu'en raison de la courte longueur des dialogues, l'Assistant de Conversation n'a pas toujours la possibilité de parler, expliquant que ce nombre soit inférieur à 1. Elle diminue avec un SER croissant mais reste toujours positive, même avec beaucoup de bruit. Cela signifie que le système a appris à agir uniquement avec un score de confiance suffisamment élevé, ce qui est un comportement souhaitable pour l'AC. En effet, dans le cas où il n'est pas sûr d'avoir compris, il préfère laisser faire la conversation plutôt que de la perturber inutilement.

6.6 Conclusion

Nous avons vu dans ce chapitre une troisième manière de considérer l'humain dans le cas du dialogue humain/machine, ou plus précisément ici dans le cas du dialogue humain/humain assisté par ordinateur. Ici, la modélisation traditionnelle du dialogue humain/machine n'est plus applicable car il n'est pas suffisant de chercher à optimiser le résultat final du dialogue. Le dialogue humain/humain doit garder sa fluidité et son cours naturel et de fait, il est nécessaire de contrebalancer l'optimisation directe de celui-ci par un coût mesurant la perturbation induite par les agissements de l'assistant de conversation.

Comme nous l'avons vu cependant, utiliser la méthode présentée dans Abbasi-Yadkori et al. (2015) pour résoudre les LMDP requiert de connaître les probabilités de transition P_0 et P_a. Dans notre expérience, nous avons profité du fait que celle-ci était simulée pour les estimer avec la méthode de Monte-Carlo. Au chapitre suivant, nous allons présenter une méthode plus efficace permettant de calculer ces probabilités.

Chapitre 7

Un modèle de dialogue humain/humain

Dans le chapitre précédent, nous avons abordé le problème du dialogue humain/humain assisté par ordinateur. Nous avons vu que le cadre des processus de décision markoviens linéaires était adaptés à cette problématique. Selon ce paradigme, la récompense que cherche à optimiser l'assistant de conversation est un compromis entre les préférences de son utilisateur et une mesure de l'intrusivité de l'assistant de conversation dans le dialogue. Nous avons établi qu'un dialogue pouvait être vu comme un générateur d'états, échantillonnant ceux-ci à partir d'une distribution P_0 tandis que l'action d'un assistant de conversation permettait d'échantillonner l'état suivant selon une seconde distribution P_a. Nous avons établi que la divergence de Kullback-Leibler entre ces deux distributions était un choix raisonnable pour quantifier cette intrusivité.

Pour résoudre cette tâche cependant, nous avons vu qu'il était nécessaire de connaître les distributions P_0 et P_a. Nous avons proposé dans la partie expérimentale d'estimer ces distributions par comptage directement à partir de la méthode Monte-Carlo. Cependant, cette méthode présente l'inconvénient de ne fonctionner que sur un problème simulé. En effet, il n'est pas possible dans le cas d'une interaction avec des humains d'annuler une action pour voir ce qu'il se serait passé si on avait agit d'une manière différente. Par ailleurs, il n'est pas possible de généraliser les probabilités apprises à des situations non rencontrées. Dans ce chapitre, nous proposons une méthode générale pour apprendre ces distributions de probabilité.

7.1 Motivations

Que ce soit pour le dialogue humain/humain assisté par ordinateur ou le dialogue humain/machine, être capable de fournir un modèle de la conversation semble être une étape importante dans la résolution de ces problèmes. Comme nous l'avons vu précédemment, un tel modèle est nécessaire à la résolution du problème du dialogue humain/humain assisté par ordinateur par le biais des processus de décisions markoviens linéaires. Un tel modèle n'est cependant pas uniquement utile dans ce cadre. Ainsi dans le cas du dialogue humain/machine, avoir un tel modèle permet d'optimiser la politique du gestionnaire de dialogue avec des algorithmes de planification, connus pour leur efficacité (LaValle (2006)).

Traditionnellement, le dialogue est modélisé par des chaînes de Markov cachées, présentées en Figure 7.1 (Stolcke et al. (2000)). Selon cette approche, le dialogue est vu

comme un processus stochastique transitant d'états de dialogue en états de dialogue. Mais contrairement à la modélisation par les chaînes de Markov, il n'est pas supposé ici que ces états sont accessibles par un utilisateur extérieur. Lors de la transition cependant, un symbole observable est émis, donnant des renseignements sur l'état de dialogue actuel.

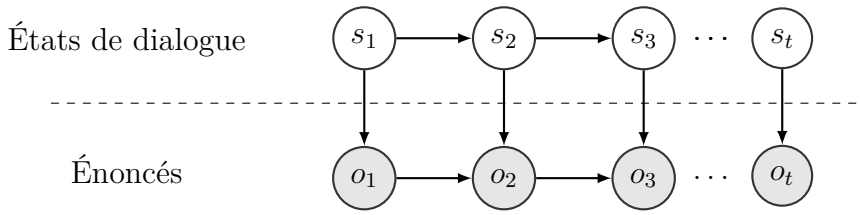


FIGURE 7.1 – Modèles de Markov cachés

Dans le cas du dialogue, les probabilités de transition entre états et les probabilités d'émission d'énoncés dans chaque état de dialogue sont inconnues. Apprendre ces probabilités est un problème connu sous le nom d'inférence. Pour cela, la procédure traditionnelle est la suivante : un premier modèle est tout d'abord implémenté manuellement par un expert humain (Alexandersson and Reithinger (1997)). Les paramètres de ce modèle sont ensuite ajustés en maximisant la probabilité du modèle avec des algorithmes tels que Markov Chain Monte-Carlo (MCMC) ou Baum-Welch/Espérance-Maximisation (EM) (Bangalore et al. (2008), Paul (2012), Zhai and Williams (2014)). Ces algorithmes reposent sur une alternance entre les étapes d'optimisation et de mise à jour. Trois problèmes découlent cependant de cette approche. Tout d'abord, concevoir le modèle initial est un problème coûteux, très dépendant du dialogue considéré et sensible aux erreurs de modélisation humaines. Deuxièmement, les algorithmes EM et MCMC sont des procédures itératives convergeant vers les optima locaux. Enfin, ces algorithmes ne passent pas à l'échelle avec le nombre d'échantillons et le nombre d'états cachés.

Dans ce chapitre, nous proposons d'apprendre un modèle de conversation, capable de renvoyer les probabilités de transitions entre états et outrepassant les problèmes abordés au paragraphe précédent. Pour cela, nous modélisons le dialogue avec le cadre des automates à multiplicité, permettant d'utiliser la méthode des moments.

Les automates à multiplicité (en anglais *Multiplicity Automaton*, MA), introduits pour la première fois dans Schützenberger (1961), constituent une classe générale de modèles graphiques, contenant les chaînes de Markov cachées. Ils sont capables de calculer une grande variété de fonctions, comme les distributions de probabilités. Apprendre ces modèles peut se faire par le biais des algorithmes spectraux (Hsu et al. (2009), Bailly et al. (2009)). Ceux-ci sont des algorithmes basées sur la méthode des moments, une méthode statistique égalisant certains moments théoriques avec leur contrepartie empirique. Le premier intérêt de ces algorithmes est que la représentation apprise par l'algorithme est exprimée en termes de variables observables, évitant alors directement les erreurs de modélisation. Plus précisément, dans ce cas, le suivi de la conversation se fait en maintenant une distribution de probabilité sur un en-

semble de futures observations. Par ailleurs, ces algorithmes se distinguent également des algorithmes Espérance-Maximisation car ils sont extrêmement rapides et souvent accompagnés de garanties théoriques. Pour toutes ces raisons, ils ont pu être appliqués avec succès sur une grande variété de tâches de traitement du langage naturel, telles que l'apprentissage d'allocation de Dirichlet latente (Anandkumar et al. (2012)), les grammaires probabilistes sans contexte (Cohen et al. (2013)), les représentations distribuées (Belanger and Kakade (2015)) ou l'analyse syntaxique (Luque et al. (2012)).

Un problème bien connu de ces algorithmes est que, dans le cas des HMM, ils ne sont pas en mesure de fournir des estimations de paramètres valides (ils ne renvoient pas des probabilités). Ce problème est pour nous d'une importance majeure car dans le cas du dialogue humain/humain assisté par ordinateur, le modèle appris n'est pas utilisable. Nous allons cependant montrer dans ce chapitre qu'il est possible de contourner cette difficulté en modélisant le dialogue comme un cas particulier d'automate de multiplicité, appelé automate fini résiduel probabiliste (PRFA), pouvant être efficacement appris et renvoyant des probabilités (Glaude and Pietquin (2016)).

7.2 Prérequis

7.2.1 Modèles de Markov Cachés

Comme abordé précédemment, du point de vue d'un observateur extérieur, un dialogue peut être vu comme un générateur d'observations o tirées d'un ensemble Σ. En fonction de l'observateur, ces observations peuvent être des énoncés (si l'observateur est un humain), des actes de dialogue ou des mots (si l'observateur est une machine). La tâche qui nous intéresse ici est de prédire les prochaines observations en fonction de ce qui a été observé dans le passé. Formellement, on suppose que chaque observation est échantillonnée selon une distribution p :

$$p(o_0, o_1, ...) = p(o_0) \prod_{t>0} p(o_t|h_t),$$

où $h_t = \{o_0, ..., o_{t-1}\}$ est appelé *historique*.

Le calcul de cette distribution pour tous les historiques possibles est souvent impossible, un problème connu sous le nom de *malédiction de l'historique*. Une compression de ces historiques appelés *état* est donc introduite. Comme nous l'avons vu en Section 6.2, dans un contexte de dialogue, ces états contiennent souvent le dernier énoncé et des informations pertinentes sur le passé. Cette information pertinente n'est toutefois pas définie avec précision et, dans la pratique, elle est conçue à la main et dépend donc fortement du domaine d'application. De plus, lorsque les dialogues parlés sont écoutés par une machine, la correspondance entre les états et les observations peut être floue, car la reconnaissance de la parole et la compréhension du langage parlé peuvent être sujettes aux erreurs.

Ce problème est généralement abordé selon le paradigme des modèles de Markov cachés (HMM) que nous avons brièvement abordé en introduction. Formellement,

Définition 7.1. *Un HMM est un processus de Markov non contrôlé ou un 5-uplet* $\{S, \Sigma, T, O, b_0\}$ *où :*

- $S = (s_1, \ldots, s_n)$ *est un ensemble fini d'états*
- Σ *est un ensemble fini d'observations*
- $T = [T_{i,j}]_{i,j \in Q}$ *est la matrice de transitions*
- $O = [O_{o,i}]_{o \in \Sigma, i \in Q}$ *est la matrice d'observation*
- $b_0 = [b_0(i)]_{i \in Q}$ *est le vecteur de croyance initial, représentant la distribution initiale sur les états*

Le vecteur de croyance b_t (*belief vector* en anglais) au pas de temps t indique, pour chaque état, quelle est la probabilité pour que le processus se trouve dans cet état. Cet état de croyance est calculé en fonction de l'état de croyance b_{t-1} du pas de temps précédent (d'où la nécessité d'une initialisation) et de l'observation o_t reçue lors de la transition entre les états s_{t-1} et s_t. Il est intéressant de noter que ce vecteur est un vecteur d'état, sa connaissance est donc suffisante pour obtenir une représentation markovienne du dialogue.

L'équation de mise à jour permettant de calculer b_t à partir de b_{t-1} et o_t dépend des matrices de transitions et d'observations. Malheureusement, dans le cas d'un dialogue, ces matrices sont inconnues a priori. Elles doivent donc être apprises à partir de données. C'est la tâche qui nous intéresse dans ce chapitre, appelée tâche d'*inférence*. Celle-ci peut être menée à bien par des algorithmes de type Baum-Welch ou Markov Chain Monte Carlo. Cependant, il est connu que ces algorithmes nécessitent beaucoup de ressources de calcul et convergent lentement vers de mauvais optima locaux (ou nécessitent de faire de fortes hypothèses).

Dans le reste de cette section, nous présentons un autre paradigme abordant le problème d'inférence de modèles graphiques adaptés à l'apprentissage d'une représentation d'un dialogue et surpassant ce modèle traditionnel sur une grande variété de points.

7.2.2 Processus stochastiques et matrices de Hankel

Les automates à multiplicité que nous allons étudier au cours de ce chapitre sont directement reliés aux matrices de Hankel que nous allons définir maintenant.

Par souci de cohérence avec l'état de l'art, nous adoptons dans ce chapitre la terminologie et les notations suivantes. Soit Σ l'ensemble fini, appelé *alphabet*, contenant toutes les observations possibles. La concaténation entre deux éléments x et y de Σ sera notée xy. Une séquence d'observations est appelée un *mot*. L'ensemble de mots sur l'alphabet Σ est désigné par Σ^*. Le mot vide est noté ϵ et les mots sont désignés par des variables surmontées d'une barre. Le cardinal d'un ensemble Q est noté $|Q|$.

Une fonction $p : \Sigma^* \to K$, avec $K = \mathbb{R}$ ou $\mathbb{R}_+$ est appelée *série formelle*. La fonction p caractérisant le dialogue défini dans la dernière section est un type particulier de série formelle appelée *processus stochastique*. Formellement,

Définition 7.2 (Thon and Jaeger (2015)). *Un processus stochastique est une fonction* $p : \Sigma^* \to [0, 1]$ *satisfaisant :*

— $p(\epsilon) = 1$
— $\forall \overline{x} \in \Sigma^* : p(\overline{x}) = \sum_{x \in \Sigma} f(\overline{x}x)$
Un processus stochastique p définie donc des probabilités sur des séquences d'observation

Étant donné une série formelle $f : \Sigma^* \to \mathbb{R}_+$, on définit sa matrice de Hankel $H \in \mathbb{R}_+^{\Sigma^* \times \Sigma^*}$ comme la matrice bi-infinie $H = [f(\overline{xy})]_{\overline{x},\overline{y} \in \Sigma^*}$:

$$H = \begin{pmatrix} f(\epsilon) & f(a) & f(b) & f(aa) & \dots \\ f(a) & f(aa) & f(ab) & f(aaa) & \dots \\ f(b) & f(ba) & f(bb) & f(baa) & \dots \\ f(aa) & f(aaa) & f(aab) & f(aaaa) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Les lignes (resp. colonnes) de cette matrice correspondent donc aux *préfixes* (resp. *suffixes*) des mots qui seront pris comme arguments de la série formelle f.

Dans le cas d'un processus stochastique p, les arguments de sa matrice de Hankel $\overline{xy}$ correspondent à la probabilité d'obtenir la séquence $\overline{xy}$. Il est important de noter que la matrice de Hankel n'est pas un modèle du processus stochastique mais est une représentation équivalente. Connaître H est équivalent à connaître p. Puisque les deux représentations sont indifféremment interchangeables, on peut définir le rang du processus stochastique comme étant le rang de H : $\text{rang}(p) := \text{rang}(H)$. Une caractéristique remarquable de la matrice de Hankel est qu'elle contient toutes les informations concernant le processus stochastique tout en n'étant uniquement définie que sur des éléments observables. Aucun état caché n'est ici introduit.

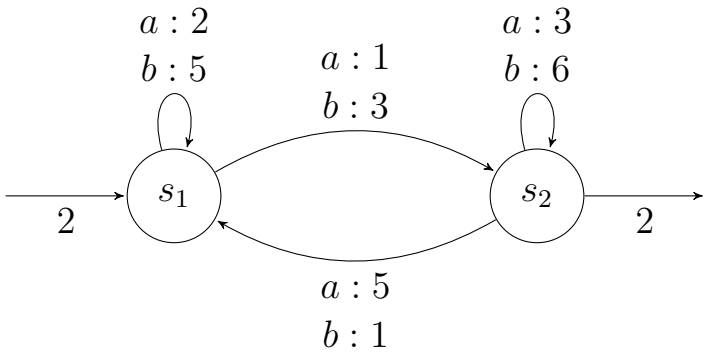
Dans la prochaine section, nous présentons les automates à multiplicité (*Multiplicity Automaton* en anglais, MA), une classe particulière de modèles graphiques reliés aux matrices de Hankel et permettant leur calcul.

7.2.3 Automates à multiplicité

Les automates à multiplicité sont une classe de modèles graphiques abstraits réalisant des séries formelles. Formellement,

Définition 7.3. *Un K-automate à multiplicité (K-MA) est un 5-uplet* $\{S, \Sigma, (A_o)_{o \in \Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty\}$ *avec :*
— S *un ensemble d'états*
— Σ *un alphabet*
— $\boldsymbol{\alpha}_0 \in K^{|S|}$ *le vecteur d'initialisation*
— $A_o \in K^{|S| \times |S|}$ *les matrices de transition*
— $\boldsymbol{\alpha}_\infty \in K^{|S|}$ *le vecteur de terminaison*
La dimension de l'automate est définie par $d = |S|$. *Soit A un K-MA réalisant une série formelle* $f_A : \Sigma^* \to K$ *définie par :*

$$\begin{aligned} \forall \overline{u} \in \Sigma^*,\ f_A(\overline{u}) &= \boldsymbol{\alpha}_0^\top A_{\overline{u}} \boldsymbol{\alpha}_\infty \\ &= \boldsymbol{\alpha}_0^\top A_{u_1} \dots A_{u_n} \boldsymbol{\alpha}_\infty \end{aligned}$$



$$\boldsymbol{\alpha}_0 = \begin{pmatrix}2\\0\end{pmatrix} \quad A_a = \begin{pmatrix}2 & 1\\5 & 3\end{pmatrix} \quad A_b = \begin{pmatrix}5 & 3\\1 & 6\end{pmatrix} \quad \boldsymbol{\alpha}_0 = \begin{pmatrix}0\\2\end{pmatrix}$$

FIGURE 7.2 – Automate à multiplicité

Passer de l'état s_1 à l'état s_2 tout en émettant le symbole b permet d'obtenir un score de 3. A priori, ce score n'a pas d'interprétation probabiliste. Dans l'hypothèse où tous les scores sont positifs, il est possibles de les interpréter comme des probabilités d'émissions de symbole (à normalisation près).

La Figure 7.2.3 illustre un automate à multiplicité. Dans le cas général, aucune hypothèse n'est faite sur les poids de l'automate et de fait, l'interprétation graphique est moins évidente que dans le cas d'un modèle de Markov caché par exemple. Mais si $K = \mathbb{R}_+$, en ignorant un facteur de normalisation, les poids peuvent être interprétés comme des probabilités d'émissions. Les automates finis non déterministes réalisant des processus stochastiques (*Non-deterministic Finite Automata realizing Stochastic Processes* en anglais, SP-NFA) sont des $\mathbb{R}_+$-MA réalisant des processus stochastiques.

Par ailleurs, il a été montré que tout HMM de d états cachés pouvait être converti en un K-MA de dimension d au plus tandis que réciproquement, un SP-NFA de dimension d pouvait ne pas être représentable par un HMM à états finis (Dupont et al. (2005), Jaeger (1998)). Le premier avantage des SP-NFA sur les HMM est donc qu'ils sont aussi expressifs que les HMM tout en étant au moins aussi compacts. En se rappelant que les HMM permettent une bonne modélisation des dialogues, nous pouvons en conclure que les SP-NFA semblent être de meilleurs modèles.

Le théorème suivant donne un ensemble de contraintes suffisantes sur les paramètres d'un $\mathbb{R}_+$-MA pour définir un SP-NFA :

Théorème 7.2.1. *(Dupont et al. (2005)) Un $\mathbb{R}_+$-MA tel que $\boldsymbol{\alpha}_0^T \mathbb{1} = 1$, $\sum_{o\in\Sigma} A_o \mathbb{1} = \mathbb{1}$ et $\boldsymbol{\alpha}_\infty^T \mathbb{1} = 1$, ou $\mathbb{1}$ est un vecteur composé de 1 et de taille $|\Sigma|$, réalise un processus stochastique et est donc un SP-NFA.*

Finalement, le lien entre matrices de Hankel et automates à multiplicité est donné par le théorème suivant :

Théorème 7.2.2. *(Carlyle and Paz (1971)) Soit f_A une série formelle réalisant un MA avec n états, alors $rank(H_{f_A}) = n$. Réciproquement, si la matrice de Hankel f*

d'une série formelle f à un rang de n, alors f peut être réalisé par un MA de n états, mais pas moins.

Ainsi, pour chaque série formelle f de rang n, il existe un MA de n états réalisant cette série formelle.

Nous venons de voir que les automates à multiplicité semblaient être de meilleurs candidats que les HMM pour donner un modèle de conversation. En ne perdant pas de vue que notre objectif est d'être capable d'estimer des matrices de transitions, nous allons voir de quelle façon il est possible d'apprendre de manière efficace ces paramètres.

7.3 Apprendre des processus stochastiques avec la méthode des moments

7.3.1 La méthode des moments

La méthode des moments est un algorithme permettant l'apprentissage des paramètres d'un automate réalisant un processus stochastique en les identifiant à ses moments empiriques. Ces algorithmes reposent principalement sur la décomposition en valeurs singulières (SVD) de la matrice de Hankel et sont souvent appelés algorithmes spectraux.

Ces algorithmes peuvent généralement être décomposés selon l'Algorithme 7.3.1.

Entrée : Un ensemble de trajectoires $\mathscr{T}$ générées par un processus stochastique p

— Trouver une base complète $\mathcal{B}$ de la matrice de Hankel H_p

— Trouver un MA de $|\mathcal{B}|$ états, vérifiant les contraintes du Théorème 7.2.1 et réalisant p

Retourner Un K-MA réalisant p

FIGURE 7.3 – Méthode des moments

Il existe dans la littérature un grand nombre de méthodes pour trouver une base complète de la matrice de Hankel. Une méthode classique est d'utiliser tous les préfixes et suffixes contenus dans l'ensemble d'entraînement. La seconde étape est celle donnant le nom de méthode des moments à ces algorithmes. Dans l'algorithme original, les matrices du Théorème 7.2.1 sont identifiées aux facteurs de la décomposition en valeurs singulières de la matrice de Hankel définie sur la base $\mathcal{B}$ (Hsu et al. (2009)). De cette deuxième étape provient le second nom donné à ces algorithmes : les algorithmes *spectraux*.

Ces algorithmes présentent cependant un inconvénient important : ils sont impropres. Cela signifie que même si l'automate appris est arbitrairement proche de la

série originale, il n'est pas nécessaire qu'ils appartiennent à la même classe. En fait, cette méthode appliquée telle quelle conduit généralement à des automates à poids négatifs. Dans ce cas, l'automate n'est pas utilisable pour les algorithmes d'apprentissage par renforcement basés sur des modèles de l'environnement. Ce problème est appelé problème des probabilités négatives (*Negative Probability Problem* en anglais, NPP).

Pour éviter le NPP, des contraintes supplémentaires doivent être insérées dans la deuxième étape de l'Algorithme 7.3.1. Dans le cas des HMM, il a malheureusement été montré que le nombre de contraintes nécessaires pour renvoyer un SP-NFA était infini **?**esposito2004). C'est pourquoi dans ce travail, nous allons étudier les SP-RFA, une classe d'automates plus petite, pouvant être efficacement apprise par la méthode des moments et adaptée aux modèles de dialogues.

7.3.2 SP-RFA

Les automates finis résiduels apprenant des processus stochastiques (en anglais *Residual Finite Automata realizing Stochastic Processes*, SP-RFA) sont des automates réalisant des processus stochastiques légèrement moins généraux que les SP-NFA (Denis and Esposito (2008)), mais pouvant être efficacement appris tout en renvoyant uniquement des paramètres positifs. Formellement,

Définition 7.4. *Un SP-RFA est un SP-NFA* $\{\Sigma, Q, \boldsymbol{\alpha}_0, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_\infty\}$ *réalisant une distribution* p *telle que pour tous les états* $q \in Q$*, il existe un mot* $\overline{u} \in \Sigma^*$ *tel que* $\boldsymbol{\alpha}_0 A_{\overline{u}} = \mathbb{1}_q$.

Les SP-RFA peuvent donc être vus comme des processus stochastiques dont la généralité se situe entre les chaînes de Markov d'ordre n (*i.e.* l'état suivant est généré conditionnellement aux n états précédents) et les HMM. Dans un contexte de dialogue, si nous supposons que le SP-RFA modélise une conversation, cela signifie que quel que soit l'état de dialogue, il y aura une séquence d'énoncés qui conduira à cet état de dialogue avec une probabilité égale à un. Cette hypothèse est raisonnable car elle est équivalente à supposer que l'état de dialogue dépend uniquement d'un historique non bruité.

Comme il a été montré que les SP-RFA peuvent être caractérisés par un ensemble fini de contraintes (Glaude et al. (2006)), il est possible de concevoir un algorithme basé sur la méthode des moments fournissant en sortie un SP-NFA.

Avant de caractériser ces contraintes, nous devons tout d'abord définir les opérateurs suivants. Soit p une série formelle et $o \in \Sigma$ un symbole. La fonction $\dot{o}p$ est la série formelle telle que $\forall u \in \Sigma$, $\dot{o}p(u) = p(ou)$. Pour tout $u \in \Sigma$ tel que $p(u\Sigma^*) > 0$, on définit p_u comme la série formelle $p_u = \frac{\dot{u}p}{p(u\Sigma^*)}$, où $u\Sigma^*$ est l'ensemble de tous les mots commençant par u. On peut alors comprendre $p_u(x)$ comme la probabilité conditionnelle de x sachant le préfixe u.

La propriété suivante donne des conditions suffisantes sur les coefficients d'un MA pour réaliser un SP-RFA :

Propriété 7.3.1. *(Glaude and Pietquin (2016)) Soit p une distribution, si il existe un ensemble de mots $\mathcal{R}$ et deux ensembles associées de nombres réels non négatifs $\{a^v_{u,o}\}_{u,v\in\mathcal{R},o\in\Sigma}$ et $\{a^v_\epsilon\}_{v\in\mathcal{R}}$ tels que $\forall u \in \mathcal{R}, \overline{p}(u) > 0$ et*

$$\forall u \in \mathcal{R}, o \in \Sigma, \dot{o}p_u = \sum_{v\in\mathcal{R}} a^v_{u,o} p_v$$
$$et\ p = \sum_{v\in\mathcal{R}} a^v_\epsilon p_v$$

alors, $\{S, \Sigma, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_0, \alpha_\infty\}$ définit un SP-RFA réalisant p, où $Q = \mathcal{R}$,

$$\begin{aligned}\boldsymbol{\alpha}_0^\top &= (a^u_\epsilon)^\top_{u\in\mathcal{R}} \forall u, v \in \mathcal{R}, A_o[u, v] \\ &= a^v_{u,o}.\end{aligned}$$

Tandis que la propriété suivante assure l'existence de ces coefficients pour un SP-RFA :

Propriété 7.3.2. *(Glaude and Pietquin (2016)) Soient $\{S, \Sigma, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty\}$ un SP-RFA et p la distribution qu'il réalise, alors il existe un ensemble de mots $\mathcal{R}$ tels que :*

$$\forall u \in \mathcal{R}, \overline{p}(u) > 0$$
$$p \in \left\{ \sum_{u\in\mathcal{R}} \alpha_u p_u \middle| \alpha_u \in \mathbb{R}_+ \right\}$$
$$\forall u \in \mathcal{R}, o \in \Sigma, \dot{o}p_u \in \left\{ \sum_{v\in\mathcal{R}} \alpha_v p_v \middle| \alpha_v \in \mathbb{R}_+ \right\}$$

Les ensembles $\{\sum_{e\in E} \alpha_e e | \alpha_e \in \mathbb{R}_+\}$ sont appelés enveloppe conique de l'ensemble E. On le note coni(E).

7.3.3 Apprendre les SP-RFA

Dans cette section, nous présentons l'algorithme CH-SP-RFA (Algorithme 7.3.3) de Glaude and Pietquin (2016) qui, étant donné un processus stochastique p généré par un SP-RFA, apprend un automate réalisant p et dont les poids correspondent à des probabilités. Cet algorithme est alors directement utilisable pour initialiser l'algorithme de Baum-Welch et affiner le modèle ou utiliser un algorithme de planification.

L'algorithme, comme la plupart des algorithmes basés sur la méthode des moments, prend en entrée une base $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ et fonctionne de la manière suivante.

Tout d'abord, on estime les séries formelles $\hat{\mathbf{p}}$ et $\dot{o}\hat{\mathbf{p}}$ (où $\mathbf{p}$ correspond à la représentation vectorielle de p). Elles seront utilisées par la suite pour apprendre les coefficients de la Propriété 7.3.1.

Ensuite, comme par hypothèse, les données ont été générées par un SP-RFA, on recherche l'enveloppe conique mentionnée en Propriété 7.3.2 dans laquelle réside notre distribution. Celle-ci peut être trouvée par exemple par un algorithme de factorisation

en matrices non-négatives. En suivant les conseils de Glaude et al. (2006), nous utilisons dans nos expériences l'algorithme de projections successives (Gillis and Vavasis (2015))

Enfin, les coefficients donnés par la Propriété 7.3.1 sont régressés dans l'enveloppe conique mentionnée précédemment, conditionnés par ailleurs avec les contraintes du Théorème 7.2.1 dans l'objectif de retourner un processus stochastique.

On peut remarquer que l'automate retourné par l'algorithme n'est pas nécessairement un SP-RFA car nous avons uniquement conditionné l'algorithme pour retourner un SP-NFA. On a supposé que les séries formelles que nous essayons d'apprendre en sont mais en pratique, il est suffisant de retourner un processus stochastique.

Algorithme 11 Algorithme CH-SP-RFA

Entrées : Une dimension cible d, une base complète séparable $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, et un ensemble de données.

Pour u $\in \hat{\mathcal{P}}$ **faire**

Estimer $\hat{\mathbf{p}}_u$ et $\dot{o}\hat{\mathbf{p}}_u$ à partir de l'ensemble d'entraînement $\hat{\mathbf{d}}_u \leftarrow (\hat{\mathbf{p}}_u^\top \ \dot{o}_1\hat{\mathbf{p}}_u^\top \ \dots \dot{o}_{|\Sigma|}\hat{\mathbf{p}}_u^\top)$

Fin pour

Trouver $\hat{\mathcal{R}}$ un ensemble de d préfixes de $\mathcal{P}$ tels que $\forall u \in \hat{\mathcal{R}}, \hat{\mathbf{d}}_u > 0$ et $\hat{\mathbf{d}} \in \text{coni}(\hat{\mathbf{d}}_u | u \in \hat{\mathcal{R}})$

Pour u $\in \hat{\mathcal{R}}$ **faire**

$$\hat{a}^v_{u,o}\} \leftarrow \underset{a^v_{u,o}}{\text{argmin}} \sum_{o\in\Sigma} \left\| \dot{o}\mathbf{p}_u - \sum_{v\in\hat{\mathcal{R}}} a^v_{u,o}\hat{p}_v \right\|_2$$
$$\text{st} \sum_{u\in\hat{\mathcal{R}}} a^u_\epsilon = 1 \text{ and } a^u_\epsilon \geq 0$$

Fin pour

$$\{\hat{a}^u_\epsilon\} \leftarrow \underset{\{a^u_\epsilon\}}{\text{argmin}} \left\| \hat{\mathbf{p}} - \sum_{u\in\hat{\mathcal{R}}} a^u_\epsilon \right\|_2$$
$$\text{st} \sum_{u\in\hat{\mathcal{R}}} a^u_\epsilon = 1 \text{ and } a^u_\epsilon \geq 0$$

$$\hat{\boldsymbol{\alpha}}_0^\top \leftarrow (\hat{a}^u_\epsilon)^\top_{u\in\mathcal{R}}$$

Pour $o \in \Sigma$ **faire**

$$\hat{A}_o \leftarrow (a^v_{u,o})_{u,v\in\hat{\mathcal{R}}}$$

Fin pour

7.4 Cadre expérimental

7.4.1 Le corpus de dialogue Ubuntu

Nos expériences ont été menées sur le corpus de dialogue Ubuntu, qui consiste en un million de dialogues dyadiques de 3 tours ou plus extraits des journaux de canaux IRC d'Ubuntu (Lowe et al. (2015)). Les dialogues de ce jeu de données suivent toujours la même structure : un utilisateur pose d'abord au salon une question, à laquelle un autre utilisateur répond.

Les étapes de prétraitement suivantes ont ensuite été appliquées sur le corpus. Tout d'abord, les expressions ont été symbolisées en utilisant le tokenizer *twokenize*[1]. Ensuite, une radicalisation et une lemmatisation ont été appliquées à l'aide de la boîte à outils NLTK (Bird (2006)). Enfin, les mots d'arrêt de NLTK et les mots apparaissant moins de 10 fois (correspondant généralement à des fautes d'orthographe) ont été supprimés.

Nous avons finalement utilisé deux types de représentations pour les énoncés qui seront décrits dans les sections suivantes : l'Allocation de Dirichlet Latente et l'Analyse Sémantique Latente, toutes deux calculées avec la bibliothèque Gensim (Rehurek and Sojka (2010)). De plus, nous avons conservé le jeton `Fin de tour` présent dans les dialogues et en avons ajouté un autre `Fin de dialogue` à la fin de chaque dialogue.

7.4.2 Représentation des énoncés

7.4.2.1 Allocation de Dirichlet latente

L'allocation de Dirichlet Latent (LDA) représente les énoncés sous forme de mélange de sujets et chaque mot de cet énoncé est généré par l'un de ces sujets (Blei et al. (2003)). Plus précisément, la LDA suppose que chaque mot d'un énoncé est généré selon la procédure suivante :

Entrée : Une allocation de Dirichlet latente
— Tirer un sujet à partir de la LDA
— Tirer un mot à partir de la distribution induite par le sujet
Retourner Un énoncé mis sous la forme d'un sac de mots

FIGURE 7.4 – Allocation de Dirichlet latente

Dans les expériences, les vecteurs de LDA considérés sont un mélange de 30 sujets.

7.4.2.2 Analyse Sémantique Latente

L'analyse sémantique latente (LSA) projette des énoncés dans un espace de concepts où deux énoncés tirés du même concept sont proches les uns des autres (Deerwester

1. https ://github.com/myleott/ark-twokenize-py

Dimension du modèle	CH-SP-RFA	Baum-Welch
10	198	879
20	200	3201
30	202	7260
40	207	-

FIGURE 7.5 – Temps de calcul pour la représentation LDA (en secondes)

et al. (1990)).

Pour parvenir à ce résultat, la LSA exécute une SVD sur la matrice de documents, définie comme la matrice contenant le nombre de mots de chaque document. Pour améliorer l'efficacité de la LSA, nous n'avons pas rempli la matrice de documents avec le nombre de mots, mais avec leurs valeurs TF-IDF, ce qui pénalise les mots fréquents et avantage les mots rares, généralement porteurs de plus d'informations.

Nous avons choisi dans les expériences de définir la LSA à partir de 100 concepts.

7.4.3 Matrice de Hankel

La matrice de Hankel a été estimée avec l'algorithme de suffixe-historique (Wolfe et al. (2005)). Cet algorithme considère tous les n-grams (avec n un paramètre de l'algorithme) comme s'il s'agissait de séquences d'apprentissage distinctes. Plus précisément, chaque n-gram est considéré comme un préfixe et on compte alors ses suffixes suivants. La matrice obtenue est finalement renormalisée pour obtenir des probabilités.

Dans les expériences, nous avons considéré des 4-grams.

7.4.4 Résultat

Pour comparer les performances des algorithmes CH-SP-RFA et Baum-Welch, nous avons choisi de comparer les log-vraisemblances négatives ainsi que les temps d'exécution.

Tout d'abord, nous avons gardé du corpus de dialogue Ubuntu uniquement les dialogues d'au moins 15 tours, conduisant à un ensemble de 257 659 dialogues dans l'ensemble d'entraînement et 4516 dialogues dans l'ensemble de test. Afin de transformer les vecteurs LDA et LSA continus en un ensemble d'observations discrètes, ils ont été clusteriser par l'algorithme des k en 30 et 50 catégories respectivement, conduisant à des alphabets de taille 32 et 52 (puisque les jetons `Fin de tour` et `Fin de dialogue` doivent être appris).

Afin d'éviter de rester dans des optima locaux, nous avons utilisé 3 redémarrages aléatoires pour l'algorithme Baum-Welch.

Les Figures 7.6 et 7.7 montrent les log-vraisemblances négatives pour la LDA et la LSA respectivement. Dans ces deux expériences, la croyance est initialisée sur les 10 premiers tours de dialogue et la vraisemblance est ensuite calculée sur la prédiction

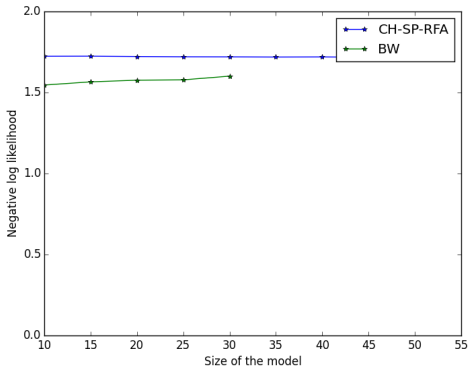


FIGURE 7.6 – Performance pour la représentation LDA

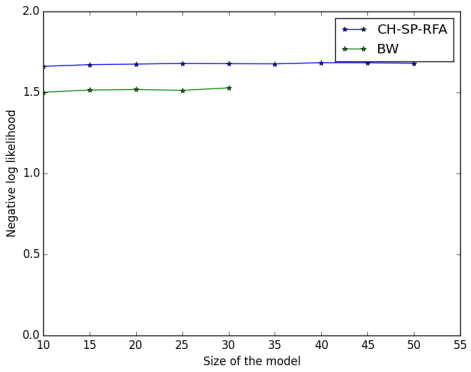


FIGURE 7.7 – Performance pour la représenation LSA

des observations suivantes. Nous n'avons pas calculé l'algorithme Baum-Welch pour les modèles dont la dimension est supérieure à 30 en raison d'un temps de calcul long. Nous observons que dans les deux expériences, les résultats des algorithmes Baum-Welch sont meilleurs que ceux de CH-SP-RFA, même si nous observons que l'ordre de grandeur est le même. Sur la représentation de la LDA, on peut noter que les performances de Baum-Welch diminuent également avec la taille du modèle, indiquant une tendance à sur-adapter.

Cependant, pour des performances comparables, la Figure 7.5 montre que la durée de fonctionnement de CH-SP-RFA est considérablement inférieure à celle de Baum-Welch. Cela peut s'expliquer par le fait que la complexité de l'échantillon est linéaire pour CH-SP-RFA car il suffit de parcourir les données une seule fois pour construire la matrice de Hankel, alors que Baum-Welch doit parcourir toutes les données à chaque itération. De plus, en raison de problèmes de convergence locale, Baum-Welch doit être exécuté plusieurs fois avec différents modèles initiaux. Enfin, la matrice de Hankel doit être calculée une seule fois pour toutes les expériences, cette étape étant de loin la plus longue de l'algorithme CH-SP-RFA et ayant duré 197 secondes.

Finalement, on observe que les petits modèles fonctionnent presque aussi bien que les plus grands. Cela peut s'expliquer par le fait que les représentations choisies ne sont pas assez riches pour comprendre complètement la signification des énoncés. Des représentations plus sophistiquées auraient pu être utilisées, mais cela aurait entraîné un plus grand alphabet et, dans ce cas, Baum-Welch aurait été totalement insoluble.

Pour ces problèmes de passage à l'échelle, nous concluons que les algorithmes spectraux, et plus particulièrement le CH-SP-RFA, en fournissant des résultats presque équivalents tout en étant plus rapides, sont une bonne alternative à l'algorithme Baum-Welch sur un grand nombre de données ou de grands alphabets.

Chapitre 8
Conclusion

La première partie de ce chapitre synthétise les différentes solutions apportées au cours de ce manuscrit à la problématique de la modélisation de l'être humain dans le dialogue humain/machine. La seconde partie propose des pistes de développements pour la conception de systèmes de dialogue.

8.1 Résumé des travaux

Les travaux réalisés au cours de cette thèse se place dans le cadre du dialogue humain/machine. Cette tâche peut être vue comme un problème de décision dans l'incertain et depuis son introduction dans Levin and Pieraccini (1997), l'apprentissage par renforcement se pose comme le paradigme de choix pour traiter ce problème, remplaçant peu à peu les systèmes de dialogue basés sur des règles.

Malheureusement, bien qu'extrêmement prometteuse, l'approche de l'apprentissage par renforcement se heurte à certaines limites. Ces limites sont d'autant plus importantes que le grand publique désire des systèmes de dialogue de type Assistant Personnel, *i.e.* des systèmes de dialogue adaptés à leur utilisateur, connaissant ses préférences et étant capable de les adapter au fur et à mesure des changements dans le comportement de son propriétaire. De nos jours, l'apprentissage par renforcement ne répond pas, ou du moins très peu à ces questions.

La thèse soutenue dans ce manuscrit est qu'une cause importante de l'existence de ces limites est la modélisation de l'utilisateur sous la forme d'un environnement stationnaire et stochastique. Nous avons abordé pour cela trois différentes situations dans lesquelles ce modèle n'est pas suffisant. Aux travers de ces situations, nous nous sommes interrogés sur le rôle de l'être humain dans les systèmes de dialogue.

Dans la première situation étudiée au Chapitre 4, nous avons montré qu'en modélisant le dialogue comme un jeu stochastique, les stratégies d'interaction sont meilleures qu'en considérant le cadre usuel des MDP. En effet, des comportements plus naturels émergent et des situations de blocage sont évitées. De plus, dans des situations ou une négociation est indispensable, les méthodes d'apprentissage par renforcement standard ne convergent pas, en raison des non-stationnarités dues à la co-adaptation intrinsèque à ce problème.

Au Chapitre 5, nous avons montré qu'il était possible d'améliorer significativement la vitesse d'apprentissage d'un algorithme d'apprentissage par renforcement hors-ligne en intégrant des conseils humains mis sous la forme d'une fonction de récompense additionnelle. Cette méthode permet également d'éviter certains comportements indésirables, pouvant négativement impacter l'expérience utilisateur. Par ailleurs, nous avons

montré que cette approche était robuste aux biais humains et que si un mauvais conseil était donné par celui-ci, le système de dialogue était à même de l'oublier. Finalement, nous avons également vu que, en raison d'une meilleure exploration de l'espace d'états, cette approche donnait de meilleurs résultats asymptotiques que l'apprentissage par renforcement classique.

Finalement, nous avons étudié au Chapitre 6 le problème du dialogue humain/humain assisté par ordinateur. Nous avons montré que, même en cherchant à impacter la conversation le moins possible, il était possible de significativement améliorer les issues de cette conversation pour l'utilisateur de l'assistant de conversation, y compris en présence de bruit. Nous avons également vu que, lorsque l'incertitude était trop grande concernant le futur de l'interaction, le système était capable de diminuer la fréquence de ses actions, par peur de négativement impacter la fluidité de celle-ci.

Pour appliquer la méthode précédente, un modèle de l'environnement est nécessaire. C'est pourquoi nous avons proposé au Chapitre 7 d'aborder ce problème en faisant appel à la Méthode des Moments, basée sur les automates à multiplicité. Nous avons alors montré que, pour des performances équivalentes à l'algorithme de Baum-Welch, le temps de calcul était significativement inférieur, rendant dès lors cette méthode plus appropriée aux problème de dialogue, dont l'une des difficultés est la gestion de la grande taille de l'espace d'états.

8.2 Pistes de recherches

Le travail réalisé dans ce manuscrit ouvre la voie a de nombreuses extensions. Tout d'abord, les expériences réalisées au Chapitre 4 ont été réalisées dans un cadre purement compétitif. Il serait intéressant de l'étendre au cas des jeux à somme générale. Pour cela cependant, un travail théorique préliminaire est nécessaire car il n'existe à l'heure actuelle pas d'algorithme de résolution efficace pour ces jeux.

Aux Chapitres 4 et 6, il a par ailleurs été supposé que les préférences de l'utilisateur du système de dialogue sont connues par celui-ci et ont été encodées sous la forme d'une fonction de récompense. En pratique cependant, cela est rarement le cas. L'apprentissage par renforcement inverse, permettant d'extraire une récompense à partir de trajectoires expertes, pourrait fournir une solution à ce problème. Une question intéressante serait de voir si, connaissant le comportement de l'utilisateur dans différents environnements, auxquels un Assistant Personnel peut avoir accès, il est possible d'extraire une fonction de récompense expliquant au mieux ces différents comportements. En suivant cette ligne de recherche, il serait également intéressant de chercher à suivre les évolutions de cette fonction de récompense, les préférences de l'utilisateur pouvant en effet changer avec le temps.

Enfin, il a été supposé au Chapitre 5 que les conseils humains étaient introduits sous la forme d'une fonction de récompense additionnelle. Mais cette manière n'est pas la seule d'introduire de l'expertise humaine. L'apprentissage par renforcement avec démonstrations expertes, par exemple, aborde le problème en considérant que l'être humain apporte des démonstrations dans des états intéressants. Fournir un cadre gé-

néral d'incorporations de connaissances (expertes ou non) dans l'apprentissage par renforcement est également une piste de travail intéressante.

Bibliographie

Y. Abbasi-Yadkori, P. L. Bartlett, X. Chen, and A. Malek. Large-scale markov decision problems with kl control cost and its application to crowdsourcing. In *Proceedings of ICMl*, 2015. (→ pages 82, 83, 84, and 88.)

N. Akchurina. Multiagent reinforcement learning : algorithm converging to nash equilibrium in general-sum discounted stochastic games. In *Proceedings of AAMAS*, 2009. (→ page 50.)

J. Alexandersson and N. Reithinger. Learning dialogue structures from a corpus. In *Proceedings of Eurospeech*, 1997. (→ page 90.)

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *Proceedings of ICML*, 2016. (→ page 30.)

A. Anandkumar, Y.-k. Liu, D. J. Hsu, D. P. Foster, and S. M. Kakade. A spectral algorithm for latent dirichlet allocation. In *Proceedings of NIPS*, pages 917–925, 2012. (→ page 91.)

Y. Ariki, T. Matsubara, and S.-H. Hyon. Latent kullback-leibler control for dynamic imitation learning of whole-body behaviors in humanoid robots. In *Proceedings of Humanoids*, 2016. (→ page 82.)

J. L. Austin. *How to do things with words*. Oxford university press, 1975. (→ pages 31 and 78.)

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv :1607.06450*, 2016. (→ page 73.)

R. Bailly, F. Denis, and L. Ralaivola. Grammatical inference as a principal component analysis problem. In *Proceedings of ICML*, 2009. (→ page 90.)

B. Bakker. Reinforcement learning with long short-term memory. In *Proceedingsof NIPS*, 2002. (→ page 72.)

S. Bangalore, G. D. Fabbrizio, and A. Stent. Learning the structure of task-driven human-human dialogs. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(7) :1249–1259, 2008. (→ page 90.)

M. Barlier, J. Perolat, R. Laroche, and O. Pietquin. Human-machine dialogue as a stochastic game. In *Proceedings of SigDial*, 2015. (→ pages 6 and 84.)

M. Barlier, R. Laroche, and O. Pietquin. A stochastic model for computer-aided human-human dialogue. In *Proceedings of Interspeech*, 2016a. (→ page 7.)

M. Barlier, R. Laroche, and O. Pietquin. Learning dialogue dynamics with the method of moments. In *Proceedings of SLT*. IEEE, 2016b. (→ page 7.)

M. Barlier, R. Laroche, and O. Pietquin. Training dialogue systems with human advices. In *Proceedings of AAMAS*, 2018. (→ pages 7, 31, and 37.)

A. G. Barto and R. S. Sutton. Goal seeking components for adaptive intelligence : An initial assessment. Technical report, DTIC Document, 1981. (→ page 16.)

D. Belanger and S. Kakade. A linear dynamical system model for text. In *Proceedings of ICML*, pages 833–842, 2015. (→ page 91.)

J. R. Bellegarda. Natural language technology in mobile devices : Two grounding frameworks. In *Mobile Speech and Advanced Natural Language Solutions*, pages 185–196. Springer, 2013. (→ page 2.)

R. Bellman. The theory of dynamic programming. Technical report, DTIC Document, 1954. (→ page 16.)

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of ICML*, 2009. (→ page 78.)

S. Bird. Nltk : the natural language toolkit. In *Proceedings of COLING*, 2006. (→ page 99.)

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3 :993–1022, 2003. (→ page 99.)

D. G. Bobrow, R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson, and T. Winograd. Gus, a frame-driven dialog system. *Artificial intelligence*, 8(2) :155–173, 1977. (→ page 2.)

G. Bourguin, A. Derycke, and J.-C. Tarby. Beyond the interface : Co-evolution inside interactive systems - a proposal founded on activity theory. In *People and Computers XV-Interaction without Frontiers*, pages 297–310. Springer, 2001. (→ page 46.)

M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2) :215–250, 2002. (→ pages 46 and 51.)

L. Breiman. *Classification and regression trees*. Routledge, 1984. (→ page 58.)

L. Buşoniu, R. Babuska, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on*, 38(2) :156–172, 2008. (→ page 50.)

J. Caelen and A. Xuereb. Dialogue et théorie des jeux. In *Congrés international SPeD*, 2011. (→ page 46.)

J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1) :26–40, 1971. (→ page 94.)

S. Chandramohan, M. Geist, F. Lefèvre, and O. Pietquin. User simulation in dialogue systems using inverse reinforcement learning. In *Proceedings of Interspeech*, 2011. (→ pages 40 and 53.)

S. Chandramohan, M. Geist, F. Lefèvre, and O. Pietquin. Behavior Specific User Simulation in Spoken Dialogue Systems. In *Proceedings of ITG Conference on Speech Communication*, 2012a. (→ page 53.)

S. Chandramohan, M. Geist, F. Lefèvre, and O. Pietquin. Co-adaptation in Spoken Dialogue Systems. In *Proceedings of IWSDS*, 2012b. (→ pages 46 and 53.)

L. Chen, R. Yang, C. Chang, Z. Ye, X. Zhou, and K. Yu. On-line dialogue policy learning with companion teaching. *Proceedings of EACLK*, 2017. (→ pages 62 and 65.)

S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1) :1, 2009. (→ page 66.)

H. Chinaei and B. Chaib-draa. Dialogue pomdp components (part ii) : learning the reward function. *International Journal of Speech Technology*, 17(4) :325–340, 2014. (→ page 39.)

H. H. Clark and E. F. Schaefer. Contributing to discourse. *Cognitive Science*, 13(2) : 259–294, 1989. (→ page 78.)

J. A. Clouse. On integrating apprentice learning and reinforcement learning. Technical report, Amherst, MA, USA, 1996. (→ page 66.)

J. A. Clouse and P. E. Utgoff. A teaching method for reinforcement learning. In *Proceedings of ICML*, 1992. (→ page 63.)

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. H. Ungar. Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of HLT-NAACL*, pages 148–157, 2013. (→ page 91.)

K. M. Colby, S. Weber, and F. D. Hilf. Artificial paranoia. *Artificial Intelligence*, 2 (1) :1–25, 1971. (→ page 1.)

H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. Human-computer dialogue simulation using hidden markov models. In *Proceedings of ASRU*, 2005. (→ page 40.)

H. Cuayáhuitl, S. Keizer, and O. Lemon. Strategic dialogue management via deep reinforcement learning. 2015. (→ page 42.)

L. Daubigney, M. Geist, S. Chandramohan, and O. Pietquin. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8) :891–902, 2012. (→ page 53.)

L. Daubigney, M. Geist, and O. Pietquin. Model-free pomdp optimisation of tutoring systems with echo-state networks. In *Proceedings of SIGDial*, 2013. (→ pages 37 and 78.)

H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3) :297–325, 2009. (→ page 35.)

H. De Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. Guesswhat ?! visual object discovery through multi-modal dialogue. In *Proceedings of CVPR*, 2017. (→ pages 32 and 37.)

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391, 1990. (→ page 99.)

L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, et al. Recent advances in deep learning for speech research at microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8604–8608. IEEE, 2013. (→ page 30.)

F. Denis and Y. Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 86(1-2) :41–77, 2008. ISSN 01692968. (→ page 96.)

L. M. Dermed and C. L. Isbell. Solving stochastic games. In *Proceedings of NIPS*, 2009. (→ page 50.)

B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, and L. Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of ACL*, 2017. (→ page 38.)

P. Dupont, F. Denis, and Y. Esposito. Links between probabilistic automata and hidden markov models : probability distributions, learning models and induction algorithms. *Pattern recognition*, 38(9) :1349–1371, 2005. (→ page 94.)

O. Dušek and F. Jurčíček. A context-aware natural language generator for dialogue systems. 2016a. (→ page 32.)

O. Dušek and F. Jurčíček. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. 2016b. (→ page 32.)

W. Eckert, E. Levin, and R. Pieraccini. User modeling for spoken dialogue system evaluation. In *Proceedings of ASRU*, 1997. (→ page 40.)

J. Edlund, J. Gustafson, M. Heldner, and A. Hjalmarsson. Towards human-like spoken dialogue systems. *Speech communication*, 50(8) :630–645, 2008. (→ page 25.)

I. Efstathiou and O. Lemon. Learning non-cooperative dialogue behaviours. In *Proceedings of SIGDIAL*, 2014. (→ pages 46 and 53.)

L. El Asri, R. Laroche, and O. Pietquin. Reward shaping for statistical optimisation of dialogue management. In *Proceedings of SLSP*, 2013. (→ page 65.)

L. El Asri, H. Khouzaimi, R. Laroche, and O. Pietquin. Ordinal regression for interaction quality prediction. In *Proceedings of ICASSP*, 2014. (→ page 39.)

L. El Asri, R. Laroche, and O. Pietquin. Dinasti : Dialogues with a negotiating appointment setting interface. In *Proceedings of LREC*, 2014. (→ page 46.)

L. El Asri, J. He, and K. Suleman. A sequence-to-sequence model for user simulation in spoken dialogue systems. In *Proceedings of Interspeech*, 2016a. (→ pages 40, 62, and 70.)

L. El Asri, B. Piot, M. Geist, R. Laroche, and O. Pietquin. Score-based inverse reinforcement learning. In *Proceedings of AAMAS*, 2016b. (→ pages 39, 69, and 81.)

L. El Asri, R. Laroche, and O. Pietquin. Compact and interpretable dialogue state representation with genetic sparse distributed memory. In *Proceedigs of IWSDS*, 2017. (→ page 37.)

M. S. English and P. A. Heeman. Learning mixed initiative dialog strategies by using reinforcement learning on both conversants. In *Proceedings of HLT/EMNLP*, 2005. (→ page 55.)

M. Eric, L. Krishnan, F. Charette, and C. D. Manning. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of SIGDIAL*, 2017. (→ page 38.)

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. pages 503–556, 2005. (→ pages 26 and 52.)

R. Evans and J. Gao. Deepmind ai reduces google data centre cooling bill by 40 https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/, 2016. (→ page 17.)

M. Fatemi, L. E. Asri, H. Schulz, J. He, and K. Suleman. Policy networks with two-stage training for dialogue systems. 2016. (→ page 61.)

E. Ferreira and F. Lefèvre. Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management. In *Proceedings of ASRU*, 2013. (→ page 65.)

E. Ferreira and F. Lefevre. Reinforcement-learning based dialogue system for human–robot interactions with socially-inspired rewards. *Computer Speech & Language*, 34 (1) :256–274, 2015. (→ pages 65 and 70.)

J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer, 1996. (→ pages 47 and 49.)

M. Gales and S. Young. The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3) :195–304, 2008. (→ page 30.)

M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of SigDial*, 2010. (→ page 41.)

M. Geist, O. Pietquin, and G. Fricout. Tracking in reinforcement learning. In *Proceedings of ICONIP*, 2009. (→ page 53.)

K. Georgila, C. Nelson, and D. Traum. Single-agent vs. multi-agent techniques for concurrent reinforcement learning of negotiation dialogue policies. In *Proceedings of ACL*, 2014. (→ pages 46, 53, 55, and 57.)

N. Gillis and S. A. Vavasis. Semidefinite programming based preconditioning for more robust near-separable nonnegative matrix factorization. *SIAM Journal on Optimization*, 25(1) :677–698, 2015. (→ page 98.)

H. Glaude and O. Pietquin. Pac learning of probabilistic automaton based on the method of moments. In *Proceedings of ICML*, 2016. (→ pages 91 and 97.)

H. Glaude, C. Enderli, and O. Pietquin. Spectral learning with non negative probabilities for finite state automaton. In *Proceedings of ASRU*, 2006. (→ pages 96 and 98.)

I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. (→ page 12.)

G. J. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Melon University, 1999. (→ page 52.)

A. L. Gorin, G. Riccardi, and J. H. Wright. How may i help you ? *Speech communication*, 23(1) :113–127, 1997. (→ page 78.)

S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz. Policy shaping : Integrating human feedback with reinforcement learning. In *Proceedings of NIPS*, 2013. (→ page 65.)

P. Guan, M. Raginsky, and R. M. Willett. Online markov decision processes with kullback–leibler control cost. *IEEE Transactions on Automatic Control*, 59(6) :1423–1438, 2014. (→ page 82.)

A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé. Expressing arbitrary reward functions as potential-based advice. In *Proceedings of AAAI*, 2015. (→ page 64.)

M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015. (→ page 31.)

J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of ACL*, 2015. (→ page 37.)

M. Henderson. Machine learning for dialog state tracking : A review. In *Proceedings of MLSLP*, 2015. (→ page 31.)

M. Henderson, B. Thomson, and S. Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of SIGDIAL*, 2013. (→ page 31.)

M. Henderson, B. Thomson, and J. Williams. The second dialog state tracking challenge. In *Proc. of SigDial*, 2014. (→ page 70.)

P. Henderson, K. Sinha, N. Angelard-Gontier, N. R. Ke, G. Fried, R. Lowe, and J. Pineau. Ethical challenges in data-driven dialogue systems. *arXiv preprint arXiv :1711.09050*, 2017. (→ page 62.)

P. J.-J. Herings and R. Peeters. Stationary equilibria in stochastic games : structure, selection and computation. 2000. (→ page 50.)

T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv :1704.03732*, 2017a. (→ page 17.)

T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, et al. Learning from demonstrations for real world reinforcement learning. *arXiv preprint arXiv :1704.03732*, 2017b. (→ page 65.)

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8) :1735–1780, 1997. (→ page 14.)

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. In *Proceedings of COLT*, 2009. (→ pages 90 and 95.)

P.-Y. Hsueh, J. D. Moore, and S. Renals. Automatic segmentation of multiparty dialogue. In *Proceedings of EACL*, 2006. (→ page 78.)

J. Hu and M. P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4 :1039–1069, 2003. (→ page 50.)

A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of ICASSP*, 1996. (→ page 32.)

S. Ioffe and C. Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *Proceedings of ICML*, 2015. (→ page 72.)

H. Jaeger. *Discrete Time, Discrete Valued Observable Operator Models : A Tutorial*. GMD-Forschungszentrum Informationstechnik, 1998. (→ page 94.)

F. Jelinek, L. R. Bahl, and R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *Information Theory, IEEE Transactions on*, 21(3) :250–256, 1975. (→ page 30.)

D. Jurafsky and J. H. Martin. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. MIT Press, 2009. (→ page 29.)

F. Jurčíček, B. Thomson, S. Keizer, F. Mairesse, M. Gašić, K. Yu, and S. Young. Natural belief-critic : a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In *Proceedings of Interspeech*, 2010. (→ page 41.)

H. Khouzaimi, R. Laroche, and F. Lefevre. Optimising turn-taking strategies with reinforcement learning. In *Proceedings of SigDial*, 2015. (→ pages 55 and 73.)

B. Kim, A. massoud Farahmand, J. Pineau, and D. Precup. Learning from limited demonstrations. In *Proceedings of NIPS*, 2013. (→ page 65.)

S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, and M. Henderson. The fourth dialog state tracking challenge. In *Proceedings of IWSDS*, 2016. (→ pages 78 and 80.)

A. H. Klopf. Brain function and adaptive systems : a heterostatic theory. Technical report, DTIC Document, 1972. (→ page 16.)

W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement : The TAMER framework. In *Proceedings of ICKC*, 2009. (→ page 66.)

W. B. Knox and P. Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of AAMAS*, 2010. (→ pages 62, 66, 68, and 69.)

V. R. Konda and J. N. Tsitsiklis. On actor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4) :1143–1166, 2003. (→ page 42.)

M. G. Lagoudakis and R. Parr. Value function approximation in zero-sum markov games. In *Proceedings of UAI*, 2002. (→ page 51.)

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec) :1107–1149, 2003. (→ page 27.)

R. Laroche and A. Genevay. The negotiation dialogue game. In *Proceedings of IWSDS*, 2016. (→ pages 84 and 85.)

R. Laroche, G. Putois, and P. Bretier. Optimising a handcrafted dialogue system design. In *Proceedings of Interspeech*, 2010. (→ pages 62 and 63.)

S. Larsson and D. R. Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3&4) :323–340, 2000. (→ page 36.)

S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006. (→ page 89.)

O. Lemon and O. Pietquin. Machine learning for spoken dialogue systems. In *Proceedings of Interspeech*, 2007. (→ page 45.)

E. Levin and R. Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of Eurospeech*, 1997. (→ pages 2, 17, 35, 45, 53, and 103.)

M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra. Deal or no deal ? end-to-end learning of negotiation dialogues. In *Proceedings of EMNLP*, 2017. (→ page 46.)

J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. In *Proceedings of EMNLP*, 2016. (→ pages 31, 32, 37, and 42.)

L. Li, J. D. Williams, and S. Balakrishnan. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *Proceedings of Interspeech*, 2009. (→ page 42.)

X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz. End-to-end task-completion neural dialogue systems. In *Proceedings of IJCNLP*, 2017. (→ page 42.)

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of ICML*, 1994. (→ page 50.)

M. L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of ICML*, 2001. (→ page 50.)

R. Lowe, N. Pow, I. Serban, and J. Pineau. The ubuntu dialogue corpus : A large dataset for research in unstructured multi-turn dialogue systems. *Proceedings of SigDial*, 2015. (→ page 99.)

F. M. Luque, A. Quattoni, B. Balle, and X. Carreras. Spectral learning for non-deterministic dependency parsing. In *Proceedings of EACL*, pages 409–419, 2012. (→ page 91.)

R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1-3) :251–281, 1996. (→ pages 62 and 63.)

T. Matsubara, V. Gómez, and H. J. Kappen. Latent kullback leibler control for continuous-state systems using probabilistic graphical models. 2014. (→ page 82.)

T. Meguro, Y. Minami, R. Higashinaka, and K. Dohsaka. Learning to control listening-oriented dialogue using partially observable markov decision processes. *ACM Transactions on Speech and Language Processing (TSLP)*, 10(4) :15, 2013. (→ page 78.)

J. Milnor. Games against nature. Technical report, RAND corporation, 1951. (→ page 45.)

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529–533, 2015. (→ pages 17 and 42.)

A.-r. Mohamed, G. Dahl, and G. Hinton. Deep belief networks for phone recognition. In *NIPS workshop on deep learning for speech recognition and related applications*, 2009. (→ page 30.)

S. Möller and N. G. Ward. A framework for model-based evaluation of spoken dialog systems. In *Proceedings of SIGDial*, 2008. (→ page 38.)

G. Neu and V. Gómez. Fast rates for online learning in linearly solvable markov decision processes. *arXiv preprint arXiv :1702.06341*, 2017. (→ page 82.)

G. Neu and C. Szepesvári. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2-3) :303–337, 2009. (→ page 38.)

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations : Theory and application to reward shaping. In *Proceedings of ICML*, 1999. (→ pages 62 and 64.)

A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of ICML*, 2000. (→ pages 38 and 81.)

A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006. (→ page 17.)

J. P. Olive and N. Spickenagel. Speech resynthesis from phoneme-related parameters. *The Journal of the Acoustical Society of America*, 59(4) :993–996, 1976. (→ page 32.)

OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018a. (→ page 17.)

OpenAI. Learning montezuma's revenge from a single demonstration. `https://blog.openai.com/openai-five/`, 2018b. (→ page 17.)

M. J. Paul. Mixed membership markov models for unsupervised conversation modeling. In *Proceedings of EMNLP*, 2012. (→ page 90.)

B. Peng, X. Li, L. Li, J. Gao, A. Celikyilmaz, S. Lee, and K.-F. Wong. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of EMNLP*, 2017. (→ page 37.)

J. Perolat, B. Piot, B. Scherrer, and O. Pietquin. Approximate dynamic programming for two-player zero-sum markov games. In *Proceedings of ICML*, 2015. (→ pages 46, 51, and 52.)

J. Pérolat, B. Piot, M. Geist, B. Scherrer, and O. Pietquin. Softened approximate policy iteration for markov games. In *Proceedings of ICML*, 2016. (→ page 51.)

J. Perolat, F. Strub, B. Piot, and O. Pietquin. Learning nash equilibrium for general-sum markov games from batch data. In *Proceedings of AISTATS*, 2017. (→ page 51.)

J. Pérolat, B. Piot, and O. Pietquin. Actor-critic fictitious play in simultaneous move multistage games. In *Proceedings of AISTATS*, 2018. (→ page 50.)

O. Pietquin. Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation. In *Proc of ICME*, 2006. (→ page 53.)

O. Pietquin. Inverse reinforcement learning for interactive systems. In *Proceedings of MLIS*, 2013. (→ page 39.)

O. Pietquin and T. Dutoit. Dynamic bayesian networks for nlu simulation with applications to dialog optimal strategy learning. In *Proceedings of ICASSP*, 2006a. (→ pages 39 and 40.)

O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2) :589–599, 2006b. (→ page 39.)

O. Pietquin and H. Hastie. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(01) :59–73, 2013. (→ pages 39, 40, and 53.)

O. Pietquin and S. Renals. Asr system modeling for automatic evaluation and optimization of dialogue systems. In *Proceedings of ICASSP*, 2002. (→ page 39.)

O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3), 2011a. (→ pages 46 and 57.)

O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3) :7, 2011b. (→ page 41.)

O. Pietquin, M. Geist, S. Chandramohan, et al. Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences. In *Proceedings of IJCAI*, 2011c. (→ page 42.)

B. Piot, M. Geist, and O. Pietquin. Boosted bellman residual minimization handling expert demonstrations. In *Proceedings of ECML*, 2014a. (→ page 65.)

B. Piot, O. Pietquin, and M. Geist. Predicting when to laugh with structured classification. In *Proceedings of Interspeech*, 2014b. (→ page 38.)

M. L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, 2014. (→ page 17.)

G. Putois, R. Laroche, and P. Bretier. Online reinforcement learning for spoken dialogue systems : The story of a commercial deployment success. In *Proceedings of SIGDIAL*, 2010. (→ page 63.)

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2) :257–286, 1989. (→ page 30.)

L. R. Rabiner and R. W. Schafer. Introduction to digital speech processing. *Foundations and trends in signal processing*, 1(1) :1–194, 2007. (→ pages 30 and 32.)

J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of ICML*, 1998. (→ page 63.)

R. Rehurek and P. Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010. (→ page 99.)

M. Riedmiller. Neural fitted q iteration-first experiences with a data efficient neural reinforcement learning method. In *Proceedings of ECML*, 2005. (→ pages 26 and 52.)

M. Riou, B. Jabaian, S. Huet, and F. Lefèvre. Joint on-line learning of a zero-shot spoken semantic parser and a reinforcement learning dialogue manager. *arXiv preprint arXiv :1810.00924*, 2018. (→ page 65.)

A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In *Proceedings of EMNLP*, 2011. (→ page 34.)

F. Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958. (→ page 13.)

S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of AISTATS*, 2011. (→ pages 35 and 65.)

N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of ACL*, 2000. (→ page 36.)

S. Russell, P. Norvig, and A. Intelligence. Artificial intelligence - a modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25 :27, 1995. (→ page 11.)

J. Schatzmann and S. Young. The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 17(4) :733–747, 2009. (→ page 40.)

J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2) :97–126, 2006. (→ page 62.)

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Proceedings of HLT*, 2007. (→ pages 40 and 53.)

J. Schatztnann, M. Stuttle, K. Weilhammer, and S. Young. Effects of the user model on simulation-based learning of dialogue strategies. In *Proceedings of ASRU*, 2005. (→ page 53.)

K. Scheffler and S. Young. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proceedings of NAACL Workshop on Adaptation in Dialogue Systems*, 2001. (→ page 40.)

M. P. Schützenberger. On the definition of a family of automata. *Information and control*, 4(2) :245–270, 1961. (→ page 90.)

J. R. Searle. *Speech acts : An essay in the philosophy of language*, volume 626. Cambridge university press, 1969. (→ page 31.)

I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv :1709.02349*, 2017. (→ page 37.)

B. Settles and B. Meeder. A trainable spaced repetition model for language learning. In *Proceedings of ACL*, 2016. (→ page 78.)

L. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10) :1095–1100, 1953. (→ pages 46, 47, and 49.)

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587) :484–489, 2016. (→ page 17.)

P. Y. Simard, S. Amershi, D. M. Chickering, A. E. Pelton, S. Ghorashi, C. Meek, G. Ramos, J. Suh, J. Verwey, M. Wang, and J. Wernsing. Machine teaching : A new paradigm for building machine learning systems. *arXiv preprint arXiv :1707.06742*, 2017. (→ page 78.)

S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning : Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16 :105–133, 2002. (→ page 62.)

S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker. Reinforcement learning for spoken dialogue systems. In *Proceedings of NIPS*, 1999. (→ page 53.)

S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker. Reinforcement learning for spoken dialogue systems. In *Proceedings of NIPS*, 2000. (→ pages 5, 6, and 17.)

B. F. Skinner. The behavior of organisms : an experimental analysis. 1938. (→ page 16.)

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv :1506.06714*, 2015. (→ page 34.)

A. Stent, R. Prasad, and M. Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of ACL*, 2004. (→ page 32.)

A. Stent, M. Marge, and M. Singhai. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of CICLing*, 2005. (→ page 32.)

A. Stolcke, N. Coccaro, R. Bates, P. Taylor, C. Van Ess-Dykema, K. Ries, E. Shriberg, D. Jurafsky, R. Martin, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3) :339–373, 2000. (→ pages 78 and 89.)

F. Strub, H. de Vries, J. Mary, B. Piot, A. Courville, and O. Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems. *Proceedings of IJCAI*, 2017. (→ pages 33, 37, 42, and 61.)

P.-H. Su, D. Vandyke, M. Gasic, N. Mrksic, T.-H. Wen, and S. Young. Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems. *Proceedings of SigDial*, 2015. (→ pages 65 and 69.)

P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. Young. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of SIGDial*, 2017. (→ page 42.)

G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2) :215–219, 1994. (→ page 17.)

M. Thon and H. Jaeger. Links between multiplicity automata, observable operator models and predictive state representations : a unified learning framework. *The Journal of Machine Learning Research*, 16(1) :103–147, 2015. (→ page 92.)

E. L. Thorndike. Animal intelligence. *Nature*, 58, 1898. (→ page 16.)

E. Todorov. Linearly-solvable markov decision problems. In *Proceedings of NIPS*, 2006. (→ pages 79 and 82.)

E. Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28) :11478–11483, 2009. (→ page 84.)

G. Tur and R. De Mori. *Spoken language understanding : Systems for extracting semantic information from speech*. John Wiley & Sons, 2011. (→ page 31.)

A. M. Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950. (→ page 1.)

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet : A generative model for raw audio. *arXiv preprint arXiv :1609.03499*, 2016. (→ page 33.)

A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, et al. Parallel wavenet : Fast high-fidelity speech synthesis. *arXiv preprint arXiv :1711.10433*, 2017. (→ page 33.)

H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang. Hybrid reward architecture for reinforcement learning. In *Proceedings of NIPS*, 2017. (→ page 17.)

O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv :1506.05869*, 2015. (→ page 35.)

W. Von Kempelen. *Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine*. Degen, 1791. (→ page 32.)

A. Waibel, M. Bett, M. Finke, and R. Stiefelhagen. Meeting browser : Tracking and summarizing meetings. In *Proceedings of the DARPA broadcast news workshop*, 1998. (→ page 78.)

M. Walker, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, S. Lee, S. Narayanan, K. Papineni, et al. Darpa communicator dialog travel planning systems : The june 2000 data collection. In *Proceedings of Eurospeech*, 2001. (→ page 38.)

M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Paradise : A framework for evaluating spoken dialogue agents. In *Proceedings of ACL*, 1997. (→ pages 38 and 81.)

G. Warnell, N. Waytowich, V. Lawhern, and P. Stone. Deep tamer : Interactive agent shaping in high-dimensional state spaces. *arXiv preprint arXiv :1709.10163*, 2017. (→ page 66.)

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4) :279–292, 1992. (→ page 24.)

W. Wei, Q. V. Le, A. M. Dai, and L.-J. Li. A goal-oriented neural conversation model by self-play, 2018. URL `https://openreview.net/forum?id=HJXyS7bRb`. (→ page 33.)

J. Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1) :36–45, 1966. (→ pages 1 and 78.)

T.-H. Wen, M. Gasic, D. Kim, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. 2015a. (→ page 32.)

T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*, 2015b. (→ page 32.)

T.-H. Wen, Y. Miao, P. Blunsom, and S. Young. Latent intention dialogue models. In *Proceedings of ICML*, 2017. (→ pages 31, 32, 37, and 42.)

E. Wiewiora, G. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of ICML*, 2003. (→ page 64.)

J. Williams, A. Raux, and M. Henderson. The dialog state tracking challenge series : A review. *Dialogue & Discourse*, 7(3) :4–33, 2016. (→ page 31.)

J. D. Williams. The best of both worlds : unifying conventional dialog systems and pomdps. In *Proceedings of Interspeech*, 2008. (→ pages 37, 62, and 64.)

J. D. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2) :393–422, 2007. (→ page 36.)

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992. (→ page 42.)

B. Wolfe, M. R. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *Proceedings of ICML*, 2005. (→ page 100.)

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model : A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2) :150–174, 2010. (→ page 36.)

S. Young, M. Gasic, B. Thomson, and J. D. Williams. Pomdp-based statistical spoken dialog systems : A review. *Proceedings of the IEEE*, 101(5) :1160–1179, 2013. (→ pages 32 and 45.)

H. Zen, K. Tokuda, and A. W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11) :1039–1064, 2009. (→ page 33.)

K. Zhai and J. D. Williams. Discovering latent structure in task-oriented dialogues. In *Proceedings of ACL*, 2014. (→ page 90.)

S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston. Personalizing dialogue agents : I have a dog, do you have pets too ? *Proceedings of ACL*, 2018. (→ page 34.)

T. Zhao and M. Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv :1606.02560*, 2016. (→ page 37.)

M. Zinkevich, A. Greenwald, and M. L. Littman. Cyclic equilibria in markov games. In *Proceedings of NIPS*, 2006. (→ page 50.)